

**BUDI RAHARJO
IMAM HERYANTO
ENJANG RK**

Bahasan:

Instalasi dan Konfigurasi MySQL

Instalasi dan Konfigurasi Apache

Instalasi dan Konfigurasi PHP

Mengenal PHP

Tipe Data, Variabel, dan Operator

Mengontrol Alur Program

Membuat Fungsi

Bekerja dengan Array

Membuat Form

Menggunakan String

Menggunakan File

Melakukan Upload File

Mengakses Database MySQL

Pengaturan Session

Workshop 1: Membuat Otentikasi User

Workshop 2: Membuat Buku Tamu

Referensi HTML

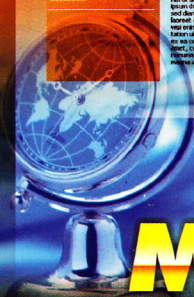
Referensi MySQL

Hi-Fi Technics Store
COMPANY SLOGAN HERE



Some New Product
Features List

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



**DISAJIKAN
SECARA
SEDERHANA
DAN
SISTEMATIS**

MODUL PEMROGRAMAN WEB

(HTML, PHP & MYSQL)

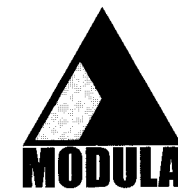


Modul

Pemrograman WEB

(HTML, PHP, & MySQL)

Budi Raharjo
Imam Heryanto
Enjang RK



Modul
Pemrograman WEB (HTML, PHP, & MySQL)
Budi Raharjo
Imam Heryanto
Enjang RK
Modula – Bandung
Telp. (022)7317812 - Fax. (022)7317896
416 hlm. : 17 x 23 cm
ISBN 978-602-8759-03-8
Copyright © 2010 - MD100004

Persembahan

Kupersembahkan buku ini untuk bapak ibuku, terima kasih atas segala kasih sayang, bimbingan, dan doanya selama ini; untuk istriku, Mira Raharjo, terima kasih atas cinta, dukungan, dan doanya; juga untuk calon permata hatiku yang selalu menjadi sumber inspirasi dan semangat hidupku.

-- Budi Raharjo

Kupersembahkan buku ini untuk kedua orang tuaku; istriku, Hernawati Heryanto; dan anak-anakku, Asfira Fauziyyah dan Rizki Dwimulyana; terima kasih atas segala kasih sayang, doa, dan dukungannya.

-- Imam Heryanto

Kupersembahkan buku ini untuk orang tuaku; istriku, Rina Julianti; dan buah hatiku, Deswita Ayuandira Dewi; yang senantiasa mendoakan dan mendukungku.

-- Enjang R.K.

Kata Pengantar

Alhamdulillah buku “Modul Praktikum Pemrograman *Web*” ini telah berhasil kami susun. Pembuatan buku ini dirancang dan disajikan untuk membantu Anda yang berkecimpung dalam bidang pemrograman *web*, baik pemula maupun bagi mereka yang ingin meningkatkan kemampuannya.

Sajian buku ini sengaja dibuat agak unik yang mungkin tidak lazim dan agak berbeda dari buku-buku lain. Buku ini dikemas dalam bentuk modul praktikum yang disertai dengan penjelasan atau keterangan kode programnya. Di sini Anda dituntut untuk mempraktikkan secara langsung materi-materi yang akan disampaikan.

Terkadang kita terlalu TAKUT untuk melakukan suatu kesalahan padahal pada saat kita melakukan kesalahan ada sebuah ilmu yang bisa kita

dapatkan. Pendekatan praktikum pada buku ini menuntun kita agar melakukan kesalahan. Dari kesalahan yang telah kita perbuat inilah kita akan memperoleh suatu pengertian, suatu pemahaman tentang suatu konsep.

Pembelajaran seperti ini menggunakan metode “pembuktian terbalik”. Bisa jadi metode ini cukup telak dan digemari oleh para pembaca. Metode ini layaknya orang yang belajar naik sepeda. Anda tidak perlu teori dan konsep yang rumit tentang bagaimana mengendarai sepeda. Sempel saja, Anda ambil sepeda, lalu coba kayuh. 99,9% Anda akan jatuh. Bangun lagi, kayuh lagi dan jatuh lagi. Lalu Anda bangun lagi, kayuh lagi, jatuh lagi, dan seterusnya sampai probabilitas Anda jatuh mendekati 0%.

Begitupun dalam buku ini, Anda disodorkan satu paket *software* untuk pembuatan *web*, yang meliputi: MySQL, Apache, dan PHP. Anda disuguhi ketiga *software* ini, lalu dituntut untuk bisa melakukan instalasi dan konfigurasi agar ketiga *software* tersebut dapat berjalan sebagaimana mestinya. Jangan takut salah! Di sini ada pembimbing, dan ada pula mentor yaitu berupa keterangan yang menjelaskan makna dan maksud dari apa yang Anda lakukan dan tuliskan.

Setelah semua *software* telah terpasang, Anda dituntut untuk mulai melakukan pemrograman menggunakan PHP. Langsung saja Anda praktikkan! Jika Anda menemui kesalahan, sudah seharusnya Anda bersyukur. Karena dengan hal tersebut Anda berarti memiliki pengalaman “pernah SALAH”. Ini berarti secara bersamaan, Anda pernah juga memiliki pengalaman BENAR (yaitu membenarkan sesuatu yang salah). Di sinilah ilmu. Di sinilah Anda mendapatkan suatu pengertian. Di sini Anda akan mendapatkan kedalaman suatu konsep.

Penulis menyuguhkan HTML dan MySQL di dalam buku ini pada bagian lampiran. Tujuannya adalah untuk menunjang keberhasilan Anda dalam

rangka mempelajari pemrograman PHP. Buku *Three-In-One* ini (HTML, MySQL, dan PHP) sengaja disusun dalam bentuk modul praktikum yang saling kait mengait sehingga para pembaca akan memperoleh pengetahuan dan pengalaman yang terintegrasi (terkait) dan komprehensif (menyeluruh/utuh) mengenai pemrograman *web*.

Selamat mencoba...

Selamat membuat kesalahan...

Selamat mendapatkan pengalaman...

Selamat berhasil dan sukses...

Semoga Anda segera mendapatkan proyek dan bisa menuntaskannya dengan benar.

Bandung, 1 Februari 2010

Penulis

Daftar Isi

Kata Pengantar	v
Daftar Isi.....	ix
Tentang Buku Ini	xv

BAGIAN 1 – INSTALASI DAN KONFIGURASI SOFTWARE

Modul 1. Instalasi dan Konfigurasi MySQL	3
Memperoleh <i>Software</i> MySQL	3
Melakukan Instalasi dan Konfigurasi MySQL.....	4
Melakukan <i>Login</i> ke <i>Server</i> MySQL.....	13
Membuat <i>User</i> di dalam MySQL	14
Memberi Hak Akses kepada <i>User</i>	15
Mencabut Hak Akses <i>User</i>	16
Menghapus <i>User</i>	17
 Modul 2. Instalasi dan Konfigurasi Apache	 19
Memperoleh <i>Software</i> Apache	19
Melakukan Instalasi Apache	20
Melakukan Konfigurasi Apache	25
Menjalankan <i>Server</i> Apache	26

Modul 3. Instalasi dan Konfigurasi PHP	29
Memperoleh <i>Software</i> PHP	29
Melakukan Instalasi PHP	30
Melakukan Konfigurasi PHP	35
Membuat Aplikasi "HelloWorld"	36

BAGIAN 2 – PHP ESSENTIAL

Modul 4. Mengenal PHP	39
Gambaran Global Aplikasi <i>Web</i>	40
Sekilas tentang PHP	41
Cara Kerja PHP	41
Sintaks Program PHP	42
Menggabungkan Kode PHP dengan Kode HTML	44
Memberikan Komentar pada Kode PHP	47
Gaya Bahasa C	47
Gaya Bahasa C++	48
Gaya Unix Shell	48
Modul 5. Tipe Data, Variabel, dan Operator	51
Pengelompokkan Tipe Data	52
<i>Integer</i>	52
<i>Floating-Point</i>	53
<i>String</i>	53
<i>Boolean</i>	54
<i>Array</i>	55
Objek	56
Pengenalan Variabel dan Cara Pembuatannya	57
Penamaan Variabel	59
Variabel Variabel	60
Referensi Variabel	61
Variabel Global dan Lokal	61
Variabel Statis	62

Parameter Fungsi	62
Konstanta	62
Operator dalam PHP	64
Operator <i>Assignment</i>	64
Operator Aritmetika	65
Operator Relasional	69
Operator Logika	72

Modul 6. Mengontrol Alur Program	73
Pemilihan Statemen	73
Menggunakan Statemen <i>if</i>	74
Menggunakan Statemen <i>switch</i>	85
Mengulang Statemen	97
Statemen <i>while</i>	97
Statemen <i>do-while</i>	102
Statemen <i>for</i>	106
Statemen <i>foreach</i>	111
Modul 7. Membuat Fungsi	115
Jenis Fungsi dalam PHP	116
Mendefinisikan dan Memanggil Fungsi	117
Fungsi Bersarang (<i>Nested Function</i>)	122
Melewatkan Parameter ke dalam Fungsi	123
<i>Pass by Value</i>	123
<i>Pass by Reference</i>	125
Membuat Parameter <i>Default</i>	127
Nilai yang Dikembalikan oleh Fungsi (<i>Return Value</i>)	128
Modul 8. Bekerja dengan Array	129
Jenis <i>Array</i> dalam PHP	130
<i>Array</i> Berindeks	130
<i>Array</i> Asosiatif	131

Membuat <i>Array</i>	131
Mengakses Elemen <i>Array</i>	132
Menggunakan Pengulangan untuk Mengakses Elemen <i>Array</i>	136
Memodifikasi <i>Array</i>	139
Mengurutkan <i>Array</i>	139
Menggunakan Fungsi <i>sort()</i>	140
Menggunakan Fungsi <i>asort()</i>	142
Menggunakan Fungsi <i>ksort()</i>	144
Menggunakan Fungsi <i>rsort()</i> , <i>arsort()</i> , dan <i>krsort()</i>	146
Modul 9. Membuat <i>Form</i>	149
Mengetahui Tag <i><form>...</form></i>	150
Atribut <i>ACTION</i>	150
Atribut <i>METHOD</i>	151
Menggunakan Metode <i>GET</i>	151
Menggunakan Metode <i>POST</i>	155
Membuat Elemen Kontrol di dalam <i>Form</i>	157
Elemen <i>TEXT FIELD</i>	157
Elemen <i>TEXT AREA</i>	160
Elemen <i>CHECK BOX</i>	163
Elemen <i>RADIO BUTTON</i>	165
Elemen <i>LIST</i>	167
Elemen <i>HIDDEN CONTROL</i>	170
Elemen <i>PASSWORD</i>	172
Modul 10. Menggunakan <i>String</i>	173
Melakukan Format <i>String</i>	174
Memperoleh Panjang <i>String</i>	176
Mencari dan Mengambil <i>Substring</i> dari Suatu <i>String</i>	177
Menghilangkan Spasi pada Bagian Awal dan Akhir <i>String</i>	180
Mengubah Huruf Kecil menjadi Besar dan Sebaliknya	183

Modul 11. Menggunakan <i>File</i>	185
Membuka dan Menutup <i>File</i>	186
Membaca Isi <i>File</i>	187
Menulis atau Menambahkan Data ke dalam <i>File</i>	192
Menggunakan <i>File</i> untuk Membuat Buku Tamu	195
Modul 12. Melakukan <i>Upload File</i>	201
Tipe <i>FILE</i> dalam Tag <i><input></i>	202
Cara Kerja dalam Proses <i>Upload</i>	203
Menentukan Lokasi <i>Upload</i>	203
Menyiapkan <i>Form</i> untuk Proses <i>Upload</i>	204
Kode PHP untuk Proses <i>Upload</i>	206
Melakukan <i>Upload</i> Beberapa <i>File</i> Sekaligus	209
Modul 13. Mengakses Database MySQL	215
Mengetahui Sekilas tentang MySQL	216
Menjalankan MySQL <i>Client</i>	216
Membuat dan Memilih <i>Database</i> yang Akan Digunakan	217
Menampilkan Informasi Metadata	218
Memanipulasi Data	219
Memersiapkan <i>Database</i> dan Tabel	220
Membangun Koneksi antara PHP dan MySQL	222
Menampilkan Data ke Halaman <i>Web</i>	223
Memasukkan Data Melalui <i>Form</i>	229
Mengubah Data dari <i>Form Web</i>	233
Menghapus Data dari <i>Form Web</i>	241
Modul 14. Pengaturan <i>Session</i>	249
Mengetahui Sistem Pengaturan <i>Session</i> dalam PHP	250
Inisialisasi yang Perlu Dilakukan	251

Langkah-langkah Menggunakan Session.....	251
Mengetahui Isi dari <i>File Session</i>	254
Melacak Nilai Variabel dari Halaman Lain.....	257
Menghapus Variabel dari Suatu Session.....	261
Mengakhiri Session.....	262

BAGIAN 3 – WORKSHOP

Modul 15. *Workshop 1: Membuat Otentikasi User* 265

Tinjauan Kasus.....	265
Membuat Tabel yang Diperlukan.....	266
Membuat Kode untuk Koneksi ke <i>Database</i>	267
Membuat <i>Form</i> Registrasi <i>User</i>	267
Membuat Kode PHP untuk Menangani Proses Registrasi.....	269
Membuat <i>Form Login</i>	271
Membuat Kode PHP untuk Menangani Proses <i>Login</i>	273
Cara Kerja Sistem.....	274

Modul 16. *Workshop 2: Membuat Buku Tamu*..... 275

Tinjauan Kasus.....	275
Membuat Tabel yang Diperlukan.....	276
Membuat Kode untuk Koneksi ke <i>Database</i>	276
Membuat Menu Buku Tamu.....	277
Membuat <i>Form</i> Entri.....	278
Membuat Kode PHP untuk Menangani Proses Entri.....	280
Membuat Kode PHP untuk Menampilkan Komentar.....	282

LAMPIRAN

Lampiran A. Referensi HTML..... 285

Lampiran B. Referensi MySQL..... 367

Tentang Buku Ini

Apa yang Dibahas dalam Buku Ini?

Buku ini merupakan buku “**3 in 1**” yang membahas PHP, HTML, dan MySQL. Adapun *web server* yang akan digunakan dalam buku ini adalah Apache.

Software-software yang akan digunakan untuk membahas materi tersebut adalah:

- **Apache 2.2.10**, yaitu perangkat yang digunakan sebagai *web server* (dalam hal ini, komputer lokal). Dengan menggunakan Apache, kode-

kode PHP yang Anda buat dapat dijalankan di dalam *web browser* di komputer lokal, tidak perlu akses internet secara langsung.

- **PHP 5**, yaitu perangkat yang digunakan untuk menerjemahkan kode-kode PHP menjadi dokumen HTML.
- **MySQL Server 5.0**, yaitu perangkat yang berperan sebagai *server database*, yang selanjutnya akan digunakan untuk mempelajari kode-kode PHP yang berkaitan atau membutuhkan akses ke *server database*.

Catatan:

Dalam mengembangkan aplikasi *web*, Anda juga dapat menggunakan paket *software* yang lain, seperti: IIS (untuk *web server*), ASP atau JSP (untuk aplikasi *web*), dan Oracle/SQL Server/Postgre (untuk *database server*).

Sampai di Mana Batasan-batasannya?

Inti pembahasan dalam buku ini adalah bagaimana Anda melakukan pemrograman *web* menggunakan PHP. Namun, untuk mendukung pemahaman tersebut Anda perlu mengetahui terlebih dahulu tentang HTML (**Lampiran A – Referensi HTML**), khususnya untuk para pemula.

Adapun materi PHP yang dibahas dalam buku ini adalah materi-materi implementatif serta materi-materi yang banyak dijumpai di lapangan.

Untuk menunjang materi PHP yang berkaitan dengan *database*, diperlukan juga pengetahuan tentang MySQL (**Lampiran B – Referensi MySQL**) sebagai salah satu *tool* yang akan digunakan sebagai *server database*. Pembahasan MySQL dibatasi pada hal-hal fundamental, seperti pembuatan *database*, pembuatan tabel, manipulasi data, dan seleksi data. Sedangkan pembahasan mengenai administrasi, seperti *backup* dan *restore database* MySQL tidak akan dikaji dalam buku ini.

Apa Tujuan dan Sasaran Buku Ini?

Tujuan dan sasaran buku ini adalah agar para pembaca memperoleh pengetahuan dan pemahaman secara UTUH dalam pembuatan aplikasi *web*, yang meliputi:

- Bagaimana aplikasi *web* bekerja (hubungan antara PHP, Apache, dan MySQL).
- Kegunaan Apache, HTML, PHP, dan MySQL serta dapat memadukannya menjadi satu aplikasi.

Untuk alasan **kemudahan**, kami penulis mempunyai inisiatif untuk menggabungkan materi-materi HTML, PHP, dan MySQL menjadi satu paket buku yang tidak terpisah pisah, kami menyebutnya sebagai buku “**3 in 1**”.

Untuk Siapa Buku Ini?

Buku ini diperuntukkan bagi:

- **Kalangan Akademisi**
 1. Guru dan Siswa SMK
Buku ini bisa dijadikan sebagai buku panduan dan buku praktikum di laboratorium komputer
 2. Dosen dan Mahasiswa
Buku ini bisa dijadikan salah satu referensi, khususnya yang berkaitan dengan kegiatan praktikum
- **Kalangan Praktisi**
Bagi individu maupun konsultan yang sering menggunakan *tools* PHP dan My SQL, serta APACHE dalam mengerjakan proyeknya.

Namun pada dasarnya buku ini diperuntukkan bagi **siapa saja**, baik para pemula maupun bagi mereka yang sudah mengenal pemrograman *web*.

Mengapa Harus Buku Ini?

Ada beberapa alasan yang bisa dijadikan landasan bagi Anda untuk memilih buku ini, di antaranya adalah:

- **Mudah dipelajari**, karena disajikan dalam bentuk praktikum.
- **Praktis**, karena bentuk buku yang “3 in 1”.
- **Simpel**, karena disajikan dalam bahasa yang sederhana dan mudah dipahami.
- **Tidak bertele-tele**, karena pembahasan materi langsung ke inti permasalahan.
- **Jelas**, karena disertai keterangan yang merupakan bedah kode program.

Bagaimana Cara Mempelajari Buku Ini?

- Bagi Anda para pemula, langkah awal yang perlu Anda lakukan adalah mempelajari terlebih dahulu materi tentang HTML (lihat **Lampiran A**).
- Setelah Anda merasa yakin telah memahami HTML, langkah berikutnya adalah melakukan instalasi *software-software* yang diperlukan (dipandu melalui buku ini).
- Jika semua *software* telah terpasang dengan benar, langkah berikutnya adalah mempelajari materi PHP, dengan cara:
 1. Kerjakan modul praktikum sesuai urutan
 2. Baca keterangan tentang kode program
 3. Lakukan analisis tentang keterkaitan antara keterangan dengan konsep

4. Bila menemui kesalahan, periksa apakah kode yang Anda tuliskan sudah benar?
5. Lakukan modifikasi program sesuai dengan kreasi yang ada dalam benak Anda

- Ketika menginjak **Modul 13** (pembahasan tentang *database*), disarankan Anda mempelajari terlebih dahulu materi tentang MySQL (lihat **Lampiran B**)

Bagian **1**

Instalasi dan Konfigurasi *Software*

Instalasi dan Konfigurasi MySQL

Memperoleh Software MySQL

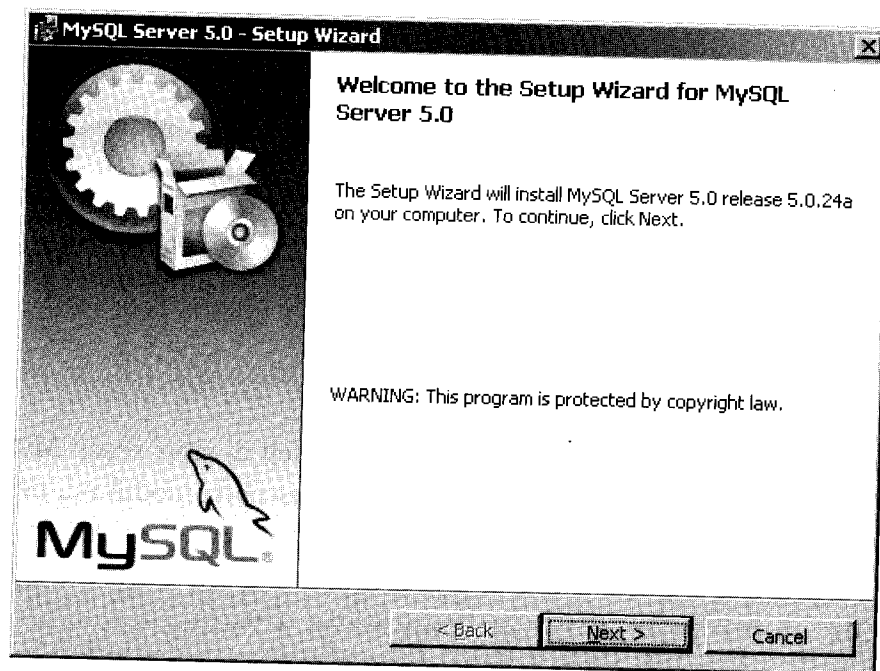
Software MySQL versi terbaru dapat Anda peroleh di situs resmi MySQL: <http://www.mysql.com/>. Dalam CD yang terdapat pada modul ini, penulis menyertakan *software* MySQL versi 5 *release* 5.0.24a. Versi inilah yang akan digunakan sebagai contoh melakukan instalasi dan konfigurasi MySQL.

Melakukan Instalasi dan Konfigurasi MySQL

Sebelum Anda mempelajari materi-materi yang terdapat pada modul ini, Anda perlu memasang *software* MySQL terlebih dahulu, yang akan kita gunakan sebagai *server database* pada saat mempelajari materi tentang akses *database* melalui PHP. Pada modul Anda akan dibimbing secara langsung untuk dapat mempraktikkan setiap materi yang disampaikan.

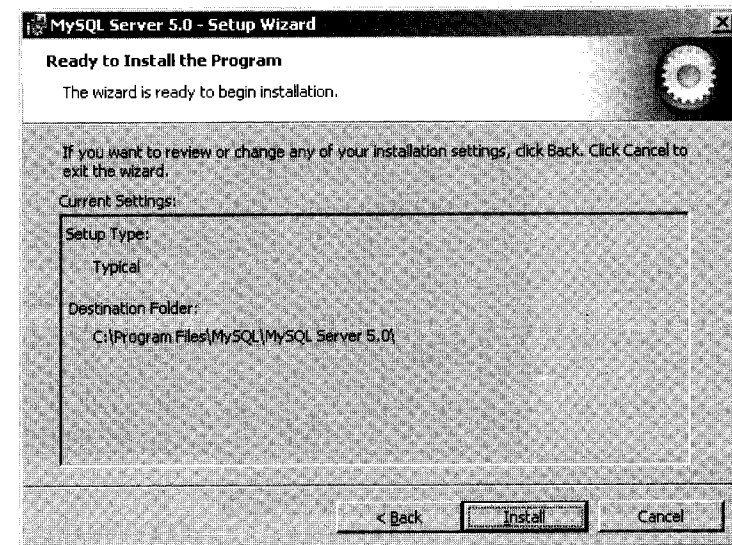
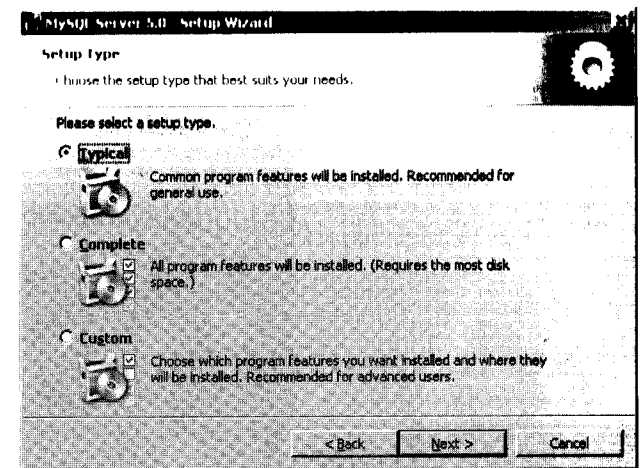
Praktikum 1-1: Melakukan Instalasi MySQL

- ✓ Masukkan CD yang Anda peroleh pada saat membeli modul ini.
- ✓ Buka direktori **MySQL**.
- ✓ Jalankan *file* `mysql-essential-5.0.24a-win32.msi`.



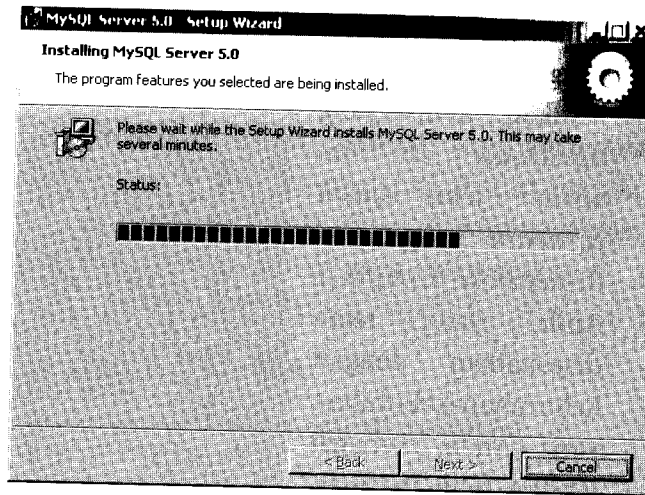
- ✓ Klik **Next** untuk beralih ke *form* selanjutnya.

- ✓ Pilih jenis instalasi yang ingin Anda lakukan. Jenis yang dimaksud di sini adalah jenis paket *software* yang akan di-instal ke komputer Anda, yang telah disesuaikan dengan kebutuhan-kebutuhan yang Anda. Sebagai latihan, pilih **Typical**, lalu klik **Next**.

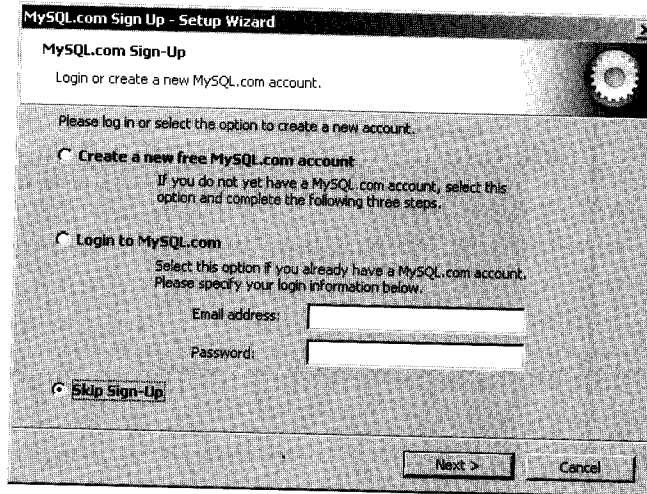


- ✓ Tujuan instalasi berada pada direktori **C:\Program Files\MySQL\MySQL Server 5.0**. Klik **Install** untuk memulai proses instalasi.

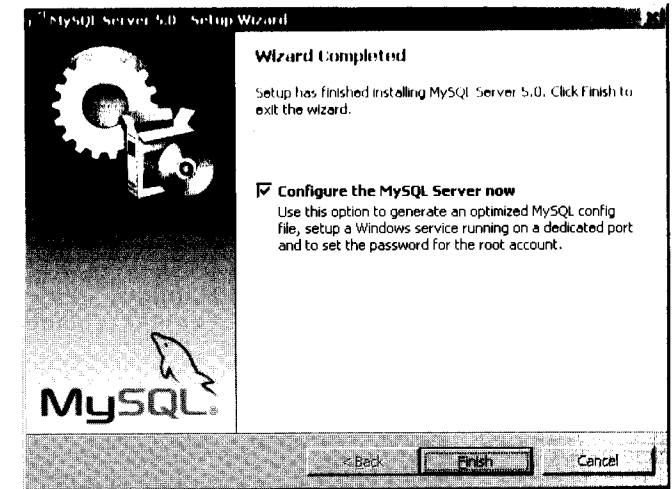
- ☑ Tunggu beberapa saat sampai muncul form seperti berikut:



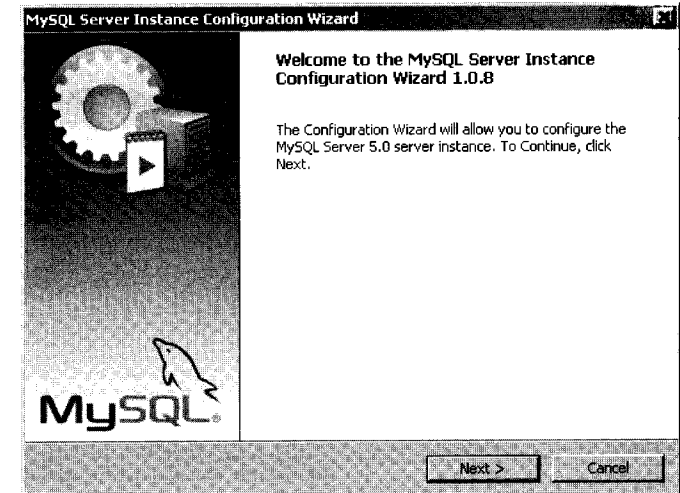
- ☑ Pilih Skip Sign-Up, lalu klik Next.



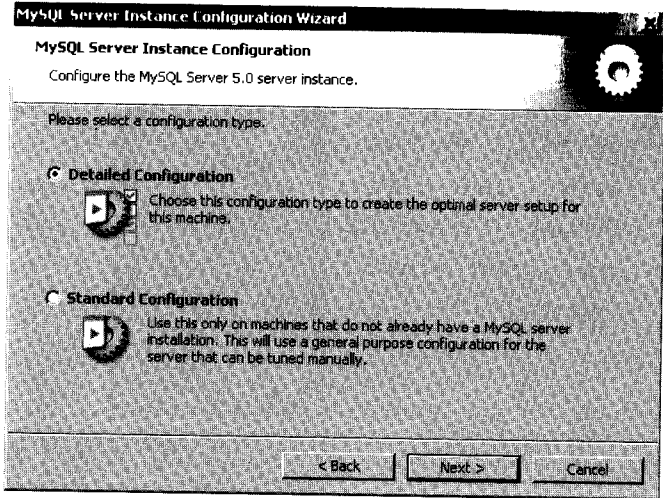
- ☑ Beri tanda *checklist* (✓) pada opsi **Configure the MySQL Server now**, lalu klik **Finish** untuk mengakhiri proses instalasi.



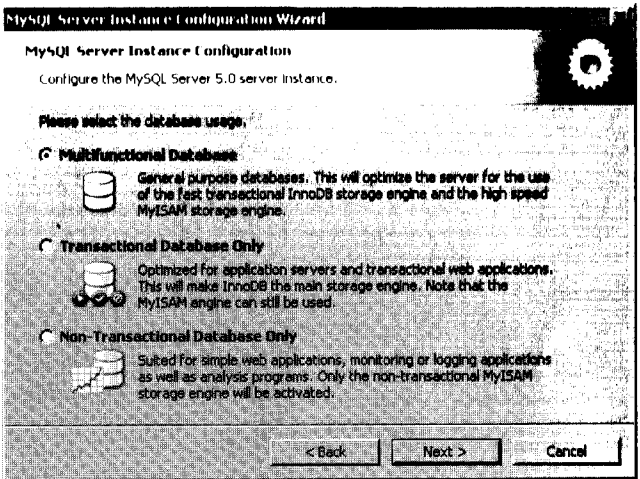
- ☑ Klik **Next** untuk melakukan konfigurasi *software*.



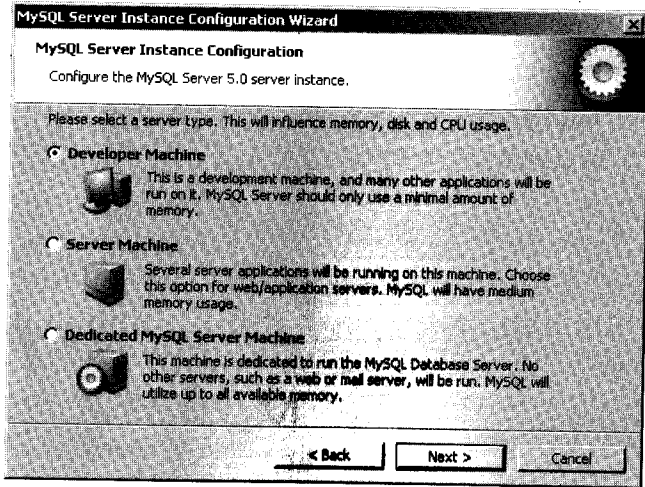
☑ Pilih **Detailed Configuration** sebagai tipe konfigurasi, lalu klik **Next**.



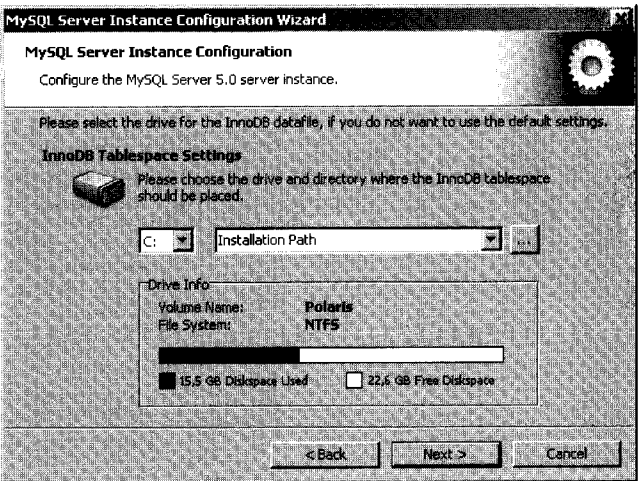
☑ Pilih **Multifunctional Database**, lalu klik **Next**.



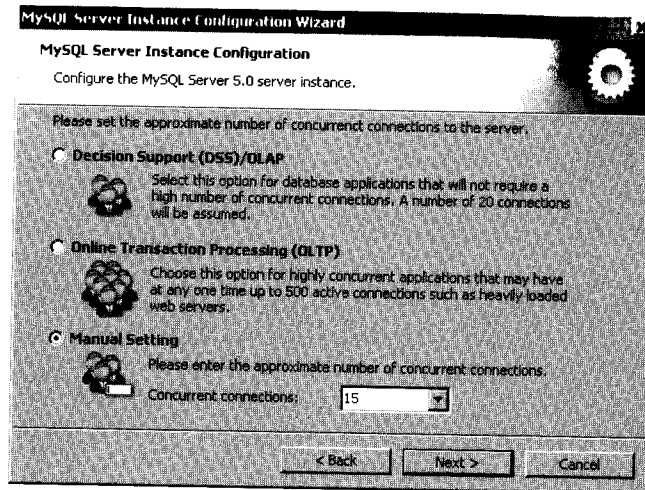
☑ Pilih **Developer Machine** sebagai tipe *server* untuk meminimalisasi penggunaan memori di komputer Anda, lalu klik **Next**.



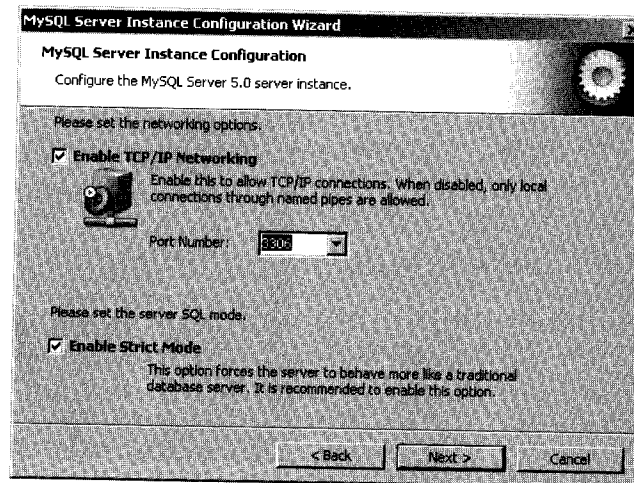
☑ Klik **Next** untuk beralih ke *form* berikutnya.



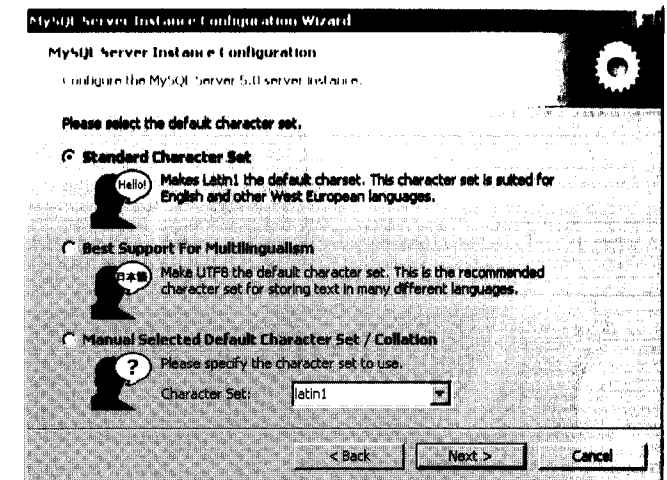
- ☒ Pilih **Manual Setting**, lalu klik **Next**.



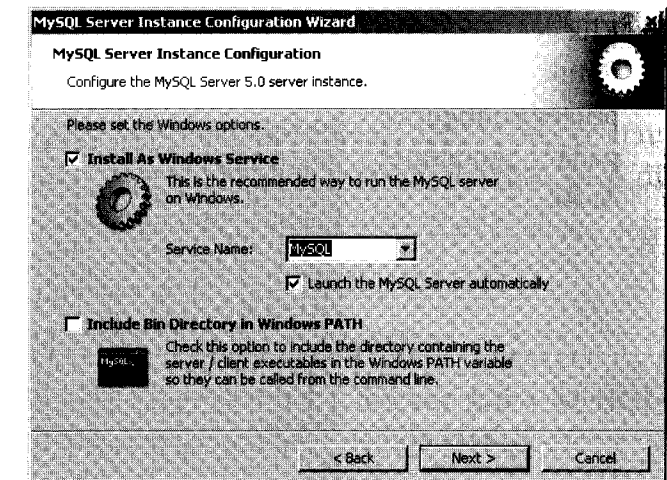
- ☒ Klik **Next** untuk melanjutkan.



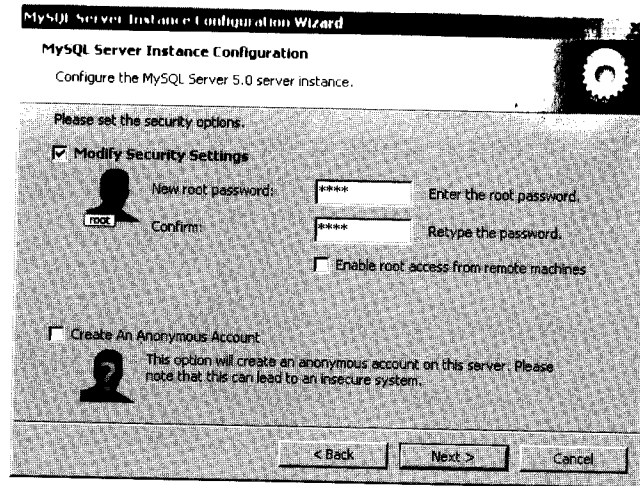
- ☒ Pilih **Standard Character Set**, lalu klik **Next**.



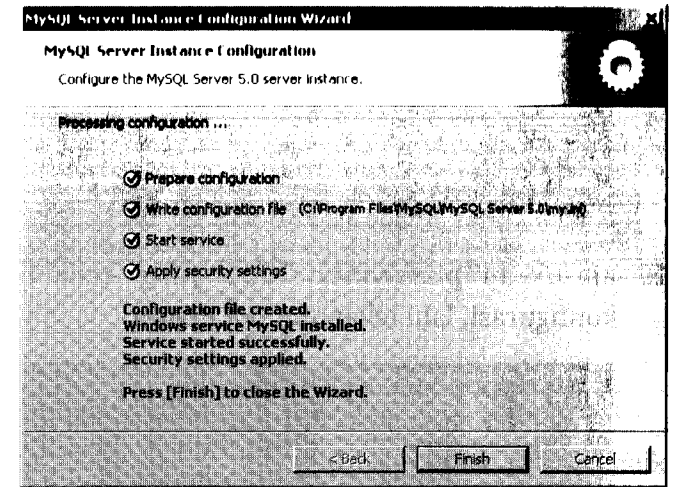
- ☒ Beri tanda *checklist* pada **Install As Window Service** dan **Launch the MySQL Server automatically** agar *server* MySQL dapat dijalankan secara otomatis pada saat komputer Anda dinyalakan.



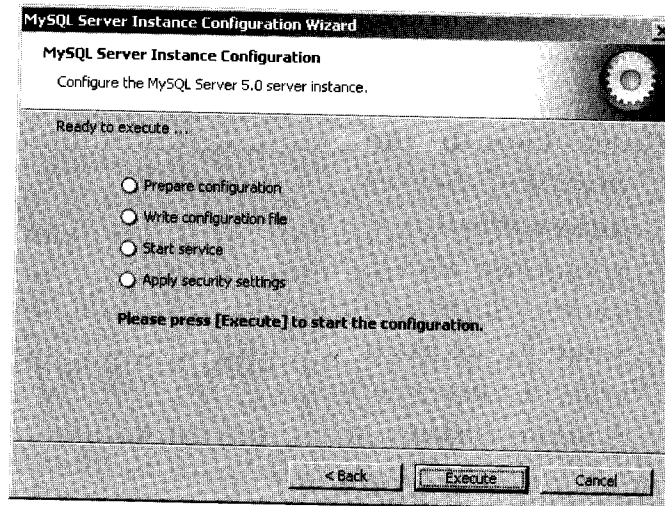
- ☑ Masukkan *password* untuk user *root*, lalu klik **Next**.



- ☑ Klik **Finish** untuk mengakhiri proses konfigurasi.



- ☑ Klik **Execute** untuk memulai proses konfigurasi pada komputer Anda.

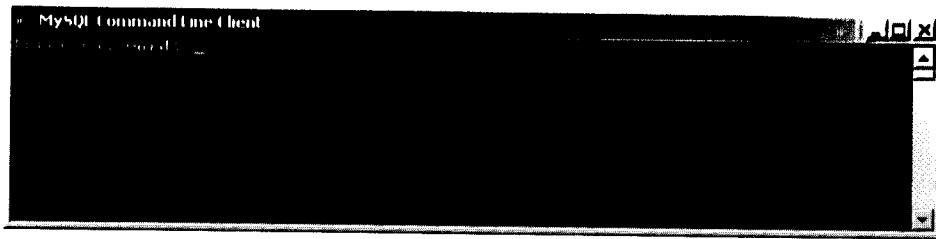


Melakukan *Login* ke Server MySQL

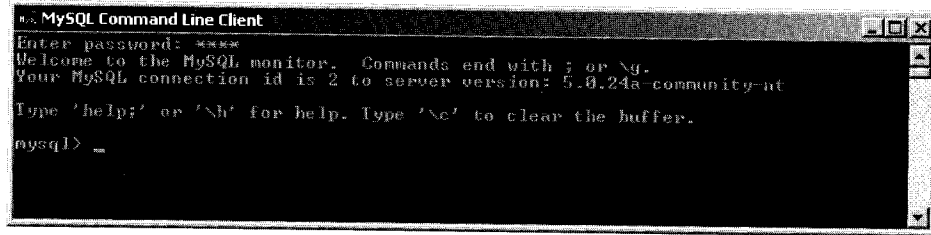
Setelah proses instalasi dan konfigurasi selesai dilakukan, Anda dapat memeriksa apakah *software* tersebut telah terpasang dengan benar atau belum melalui program *MySQL Client*. Silakan Anda praktikkan cara di bawah ini untuk melakukan *login* ke dalam *server database MySQL*.

Praktikum 1-2: Melakukan *login* ke server MySQL

- ☑ Klik **Start > Programs > MySQL > MySQL Server 5.0 > MySQL Command Line Client** untuk menjalankan program *MySQL Client*.



- ☑ Masukkan **password** sesuai yang Anda tuliskan pada saat proses konfigurasi, lalu tekan tombol **Enter**.



Jika Anda melihat *form* seperti di atas, maka *software* MySQL telah terpasang dengan benar pada komputer Anda.

Membuat User di dalam MySQL

Sebagai *administrator database*, Anda berhak menambahkan *user* ke dalam MySQL. Artinya, *user* tersebut tidak harus *login* sebagai **root**. Untuk membuat *user* di dalam MySQL, gunakan perintah berikut:

```
create user NamaUser identified by "Password";
```

NamaUser adalah nama *user* yang akan dibuat dan *Password* adalah *password* yang digunakan untuk keamanan dari *user* bersangkutan pada saat *login* ke *server database*.

Contoh:

```
create user bimo identified by "p4Nd4Wa";
```



Memberi Hak Akses kepada User

User yang baru saja ditambahkan ke dalam *server* MySQL belum memiliki hak akses apa pun. Maka dari itu, agar *user* tersebut dapat bekerja sesuai kebutuhannya, *administrator database* harus memberi hak akses kepadanya. Cara pemberian hak akses adalah:

```
grant HakAkses on NamaDatabase.NamaTabel
to NamaUser@Host
identified by "Password";
```

HakAkses adalah daftar hak yang akan diberikan kepada *user*, dan dapat berupa:

- **ALL**, semua hak akan diberikan kepada *user*
- **ALTER**, *user* dapat memodifikasi struktur tabel, kolom, maupun indeks
- **CREATE**, *user* dapat membuat *database* dan tabel
- **DELETE**, *user* dapat menghapus data dari suatu tabel
- **DROP**, *user* dapat menghapus *database* dan tabel
- **FILE**, *user* dapat mengakses (membaca dan menulis) *file*; digunakan dalam proses ekspor dan impor data
- **INDEX**, *user* dapat menambah atau menghapus indeks

- **INSERT**, *user* dapat menambah data ke dalam suatu tabel
- **PROCESS**, *user* dapat melihat dan menghentikan *system processes*
- **REFERENCES**, *user* dapat mengacu suatu kolom yang terdapat pada tabel lain; biasa digunakan dalam pembuatan *foreign key*
- **RELOAD**, *user* dapat menjalankan statemen **FLUSH**
- **SELECT**, *user* dapat melakukan seleksi data dari suatu tabel
- **SHUTDOWN**, *user* dapat melakukan *shut down* terhadap *server MySQL*
- **UPDATE**, *user* dapat melakukan modifikasi terhadap data-data yang terdapat pada suatu tabel
- **USAGE**, *user* dapat terhubung dengan *server MySQL*, tapi tidak memiliki hak akses apa pun

Contoh:

```
grant ALL on contohdb.*
to bimo@127.0.0.1
identified by "p4Nd4Wa";
```

```
mysql> grant all on contohdb.* to bimo@127.0.0.1 identified by "p4Nd4Wa";
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

Perintah di atas akan memberi semua hak akses yang terdapat pada *database contohdb* kepada **bimo**.

Mencabut Hak Akses User

Selain memberi hak akses, seorang *administrator database* juga dapat mencabut hak akses yang telah diberikan kepada seorang *user*. Caranya adalah:

```
revoke HakAkses
on NamaDatabase.NamaTabel
from NamaUser@Host;
```

Contoh:

```
revoke INSERT
on contohdb.*
from bimo@127.0.0.1;
```

```
mysql> revoke INSERT
on contohdb.*
from bimo@127.0.0.1;
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

Menghapus User

User yang telah ditambahkan ke dalam *MySQL* dapat dihapus oleh *administrator database* menggunakan perintah **DROP USER**:

```
drop user NamaUser
```

Contoh:

```
drop user bimo;
```

```
mysql> drop user bimo;
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

User yang telah dihapus tidak akan dapat melakukan *login* ke dalam server MySQL.

Instalasi dan Konfigurasi Apache

Memperoleh Software Apache

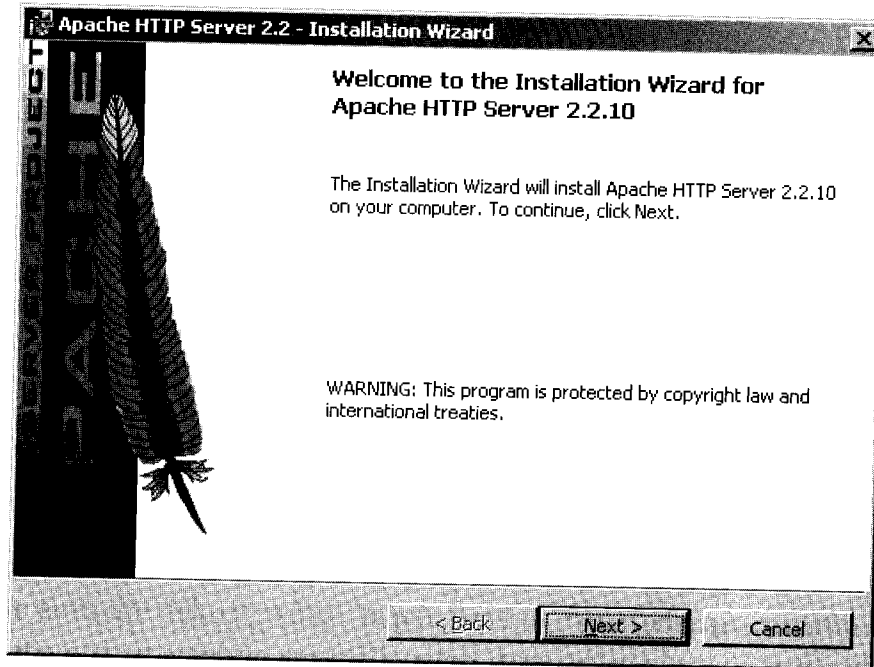
Apache adalah aplikasi *web server* yang akan kita gunakan dalam modul ini. *Software* Apache versi terbaru dapat Anda peroleh di situs resmi Apache: <http://httpd.apache.org/>. Dalam CD yang terdapat pada modul ini, penulis menyertakan *software* Apache versi 2.2.10.

Melakukan Instalasi Apache

Pada bagian ini Anda akan mempraktikkan cara melakukan instalasi Apache.

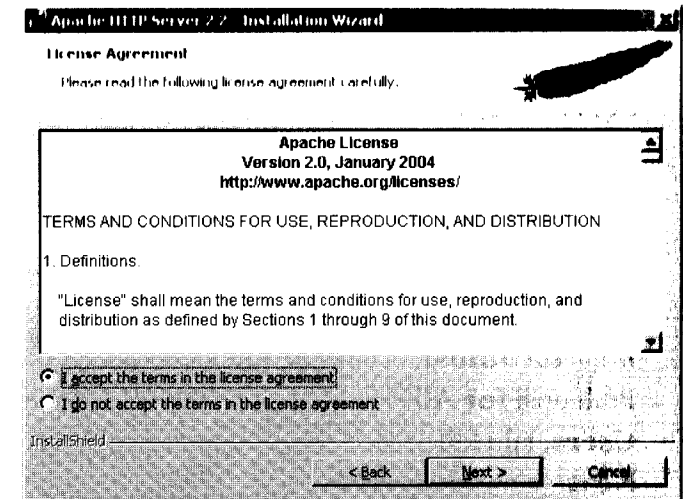
Praktikum 2-1: Melakukan Instalasi Apache

- ✓ Masukkan CD yang Anda peroleh pada saat membeli modul ini.
- ✓ Buka direktori **Apache**.
- ✓ Jalankan file **apache_2.2.10-win32-x86-openssl-0.9.8i.msi**.

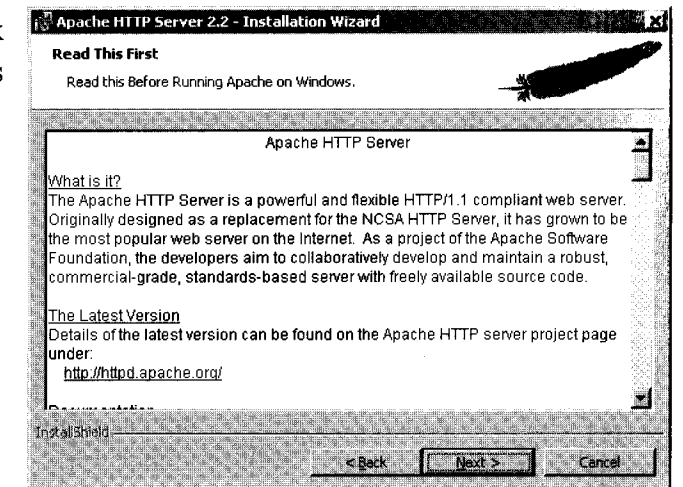


- ✓ Klik **Next** untuk beralih ke *form* selanjutnya.

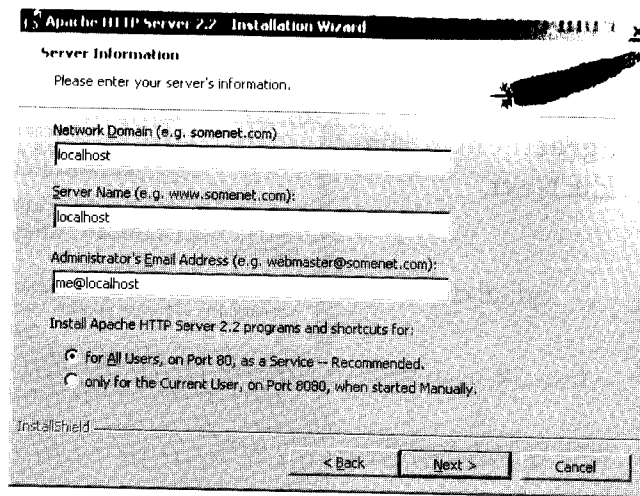
- ✓ Pilih opsi **I accept the terms in the license agreement**, lalu klik **Next**.



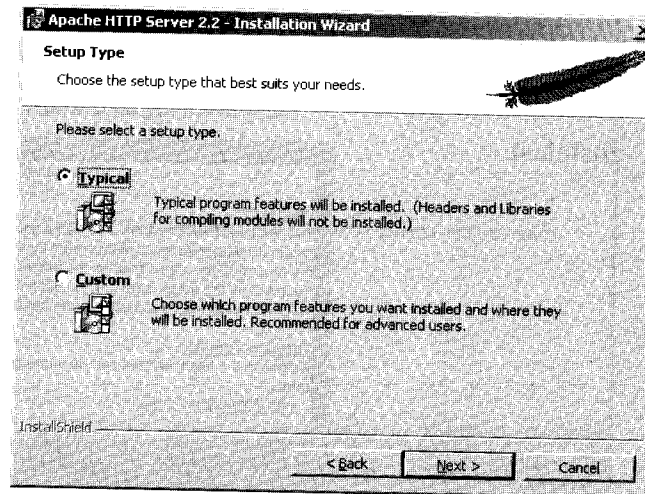
- ✓ Klik **Next** untuk melanjutkan proses instalasi.



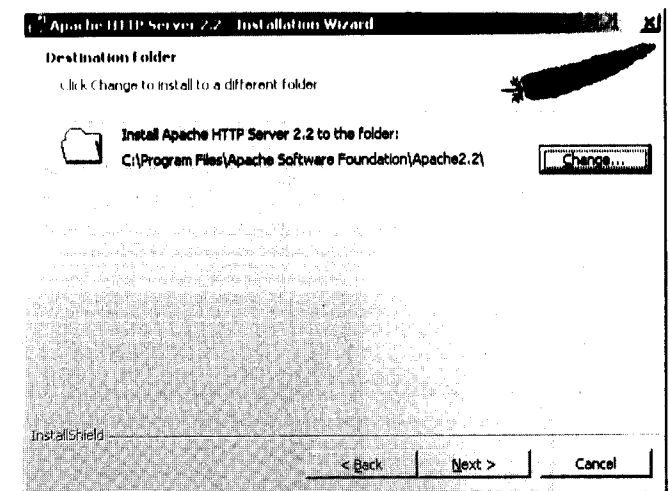
- ☑ Isikan **Network Domain** dan **Server Name** dengan **localhost** dan **Administrator's Email Address** dengan **me@localhost**. Pilih opsi **for All Users**, lalu klik **Next**.



- ☑ Pilih **Typical**, lalu klik **Next**.

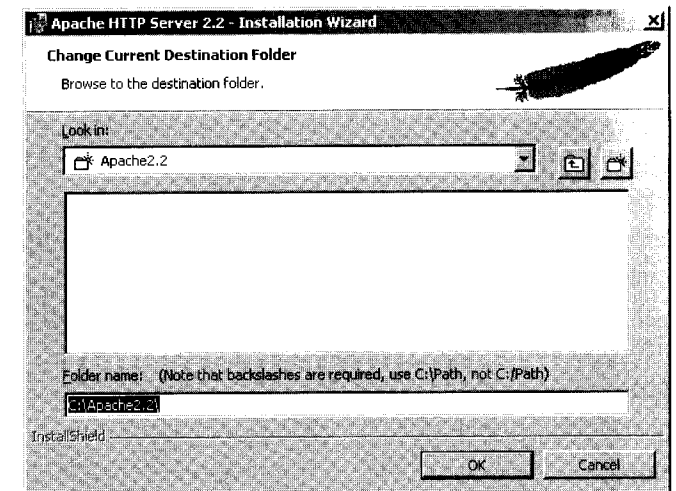


- ☑ Klik **Change** untuk menentukan lokasi instalasi.

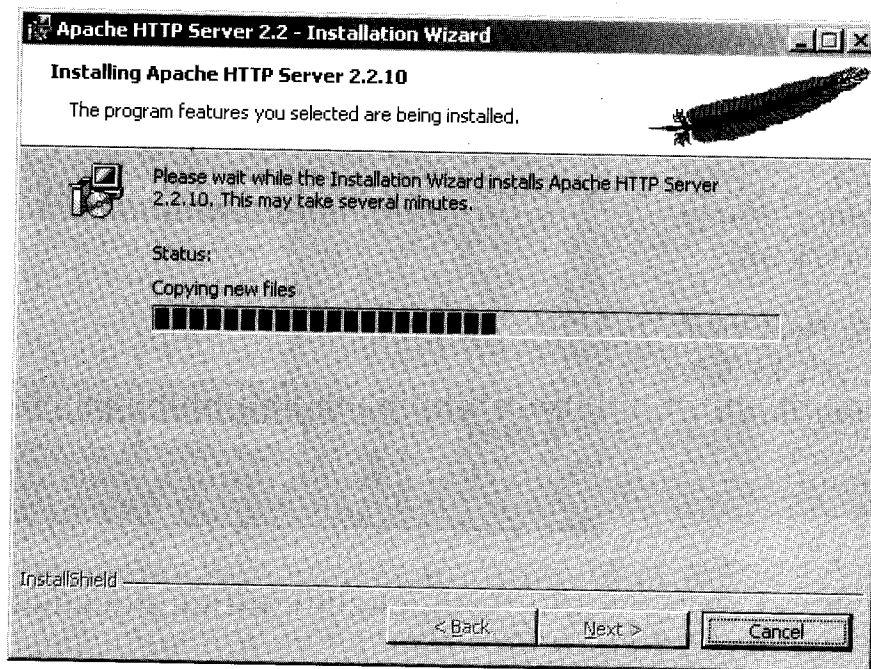
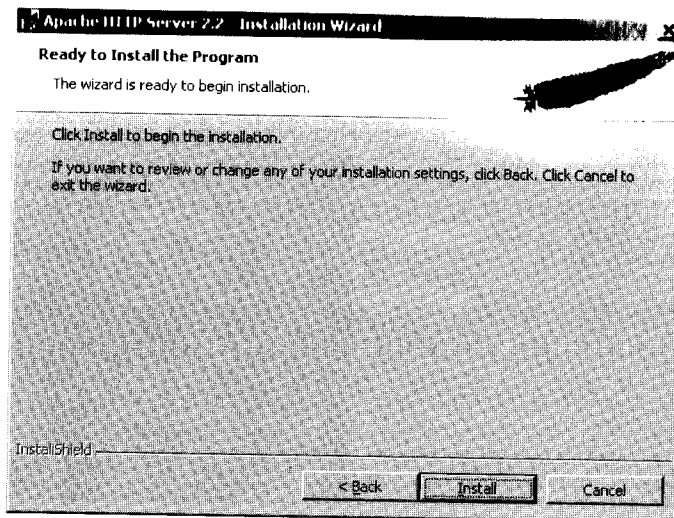


- ☑ Sebagai latihan, tuliskan **C:\Apache2.2** pada kotak **Folder Name**, lalu klik **OK**.

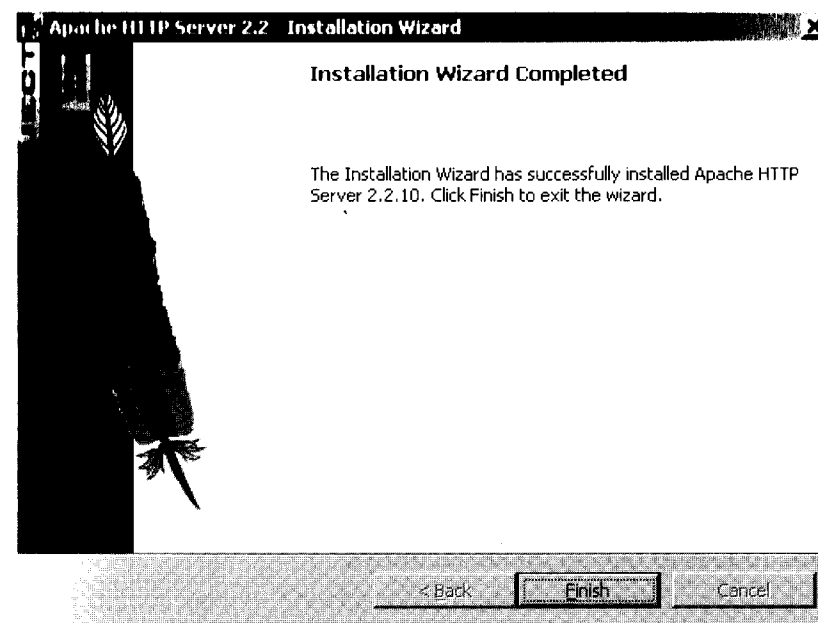
- ☑ Klik **Next** untuk beralih ke form selanjutnya.



- ☑ Klik **Install** untuk memulai proses instalasi.



- ☑ Tunggu beberapa saat sampai muncul *form* berikut:



- ☑ Klik **Finish** untuk mengakhiri proses instalasi.

Melakukan Konfigurasi Apache

Untuk dapat melaksanakan praktikum-praktikum di dalam modul ini, Anda perlu sedikit melakukan konfigurasi terhadap *software* Apache yang telah terpasang pada komputer Anda. Bagian ini akan membimbing Anda untuk mempraktikkan cara melakukan konfigurasi sehingga Apache dapat berfungsi sebagaimana mestinya.

Praktikum 2-2: Melakukan konfigurasi Apache

- ☑ Klik tombol **Start > Programs > Apache HTTP Server 2.2 > Configure Apache Server**.
- ☑ Pilih **Edit the Apache httpd.conf Configuration File**.

- ☑ Tentukan nilai untuk **DocumentRoot**:

DocumentRoot "C:/Apache2.2/htdocs"

Folder yang dituliskan di atas akan berperan sebagai tempat Anda meletakkan *file-file* kode PHP yang Anda buat.

- ☑ Ubah nilai untuk **DirectoryIndex** menjadi seperti berikut:

DirectoryIndex index.php index.html

- ☑ Carilah bagian **<IfModule mime_module>** (baris 358), tepatnya pada teks berikut:

```
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
```

Tambahkan teks di bawah ini ke dalamnya:

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php .phtml
AddType application/x-httpd-php .php3
AddType application/x-httpd-php .html
AddType application/x-httpd-php .htm
AddType application/x-httpd-php-source .phps
```

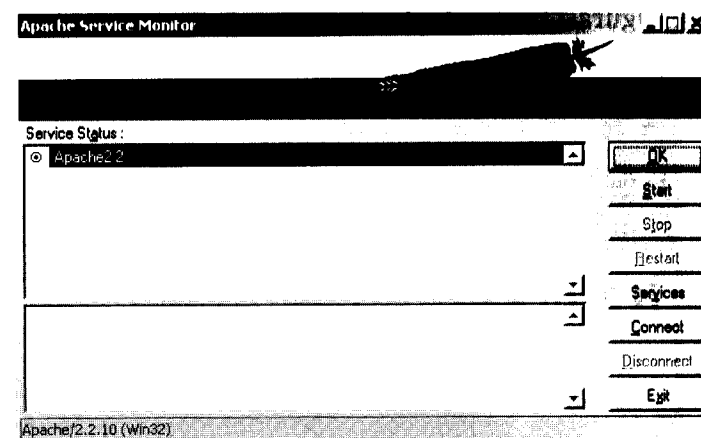
- ☑ Simpan perubahan yang telah Anda lakukan di dalam *file httpd.conf*.
- ☑ Restart Apache menggunakan **Start > Programs > Apache HTTP Server 2.2 > Control Apache Server > Restart**.

Menjalankan Server Apache

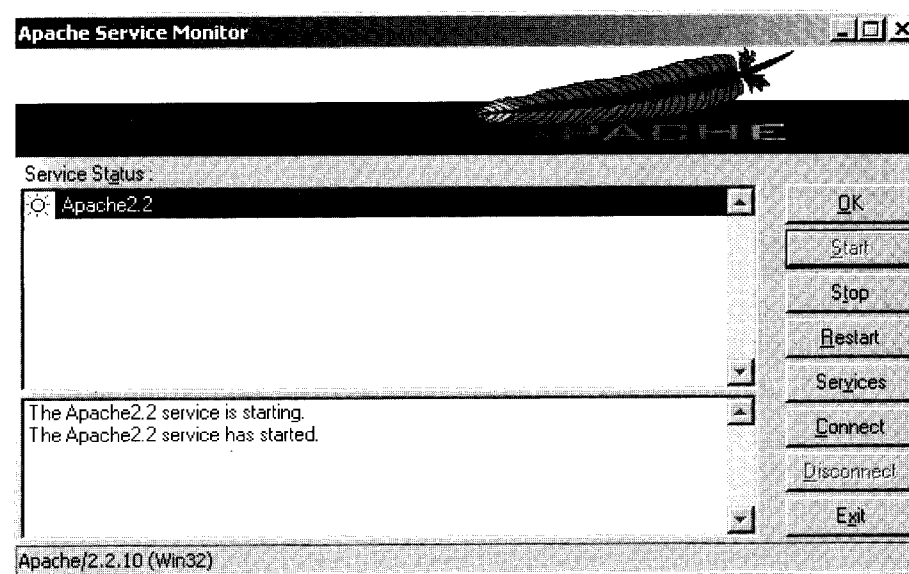
Untuk menjalankan *server* Apache, lakukan klik ganda (*double click*) pada *icon* Apache yang berada pada bagian *task bar*. Gambar *icon* Apache adalah seperti berikut:



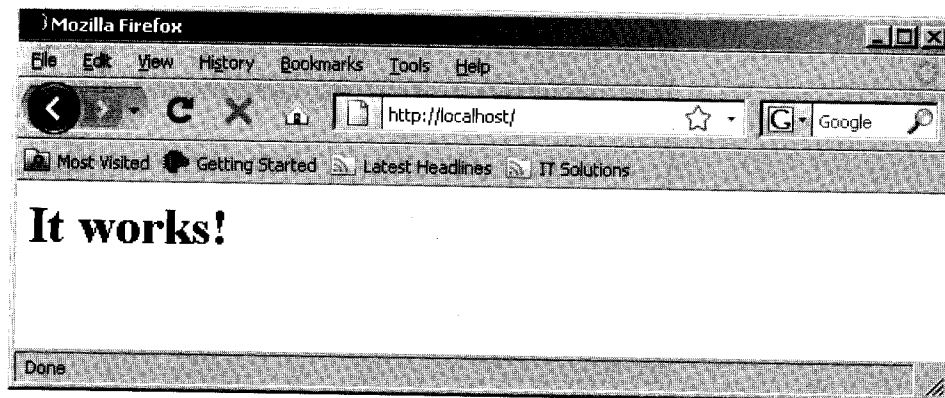
Pada saat muncul Apache Server Monitor, klik **Start**.



Indikasi bahwa *server* Apache telah diaktifkan adalah seperti pada gambar berikut:



Untuk memeriksa apakah *server* Apache siap digunakan atau belum, jalankan *web browser*, dan tuliskan alamat: **http://localhost/**. Jika konfigurasi benar, Anda akan memperoleh tampilan seperti berikut:



Instalasi dan Konfigurasi PHP

Memperoleh Software PHP

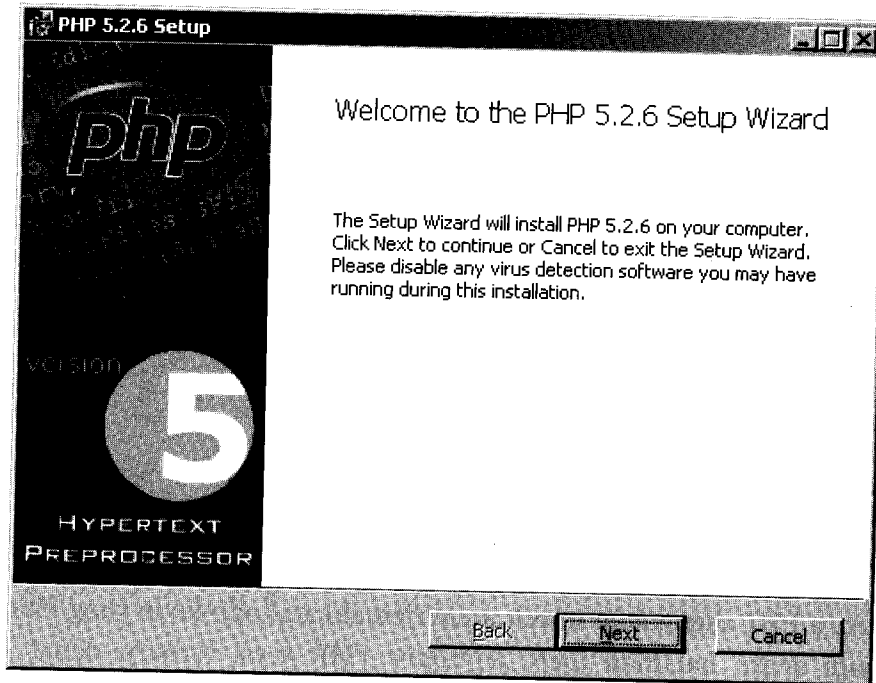
Software PHP digunakan untuk menerjemahkan kode-kode yang kita tulis menjadi suatu halaman *web*. *Software* PHP versi terbaru dapat Anda peroleh di situs resmi PHP: <http://www.php.net/>. Dalam CD yang terdapat pada modul ini, penulis menyertakan *software* PHP versi 5.2.6.

Melakukan Instalasi PHP

Pada bagian ini Anda akan mempraktikkan cara melakukan instalasi PHP.

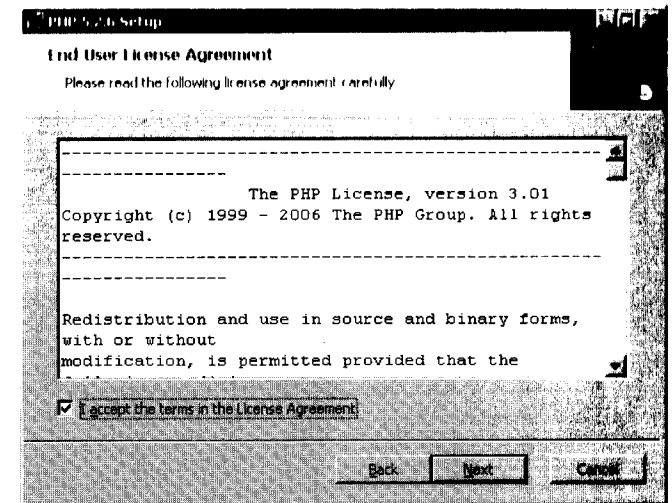
Praktikum 3-1: Melakukan Instalasi PHP

- ✓ Masukkan CD yang Anda peroleh pada saat membeli modul ini.
- ✓ Buka direktori **PHP**.
- ✓ Jalankan file **php-5.2.6-win32-installer.msi**.

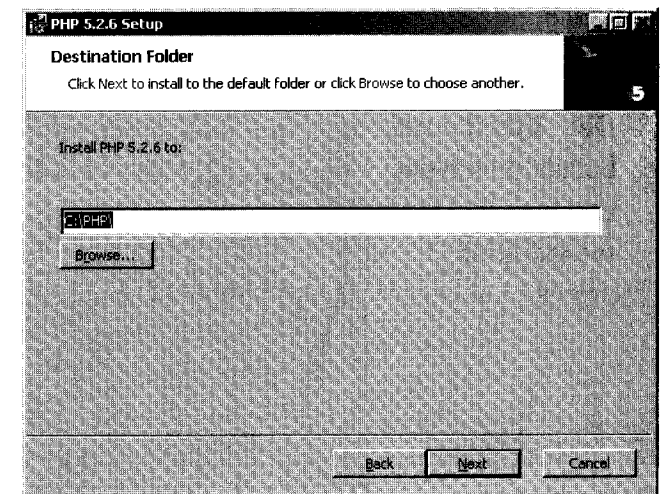


- ✓ Klik **Next** untuk beralih ke form selanjutnya.

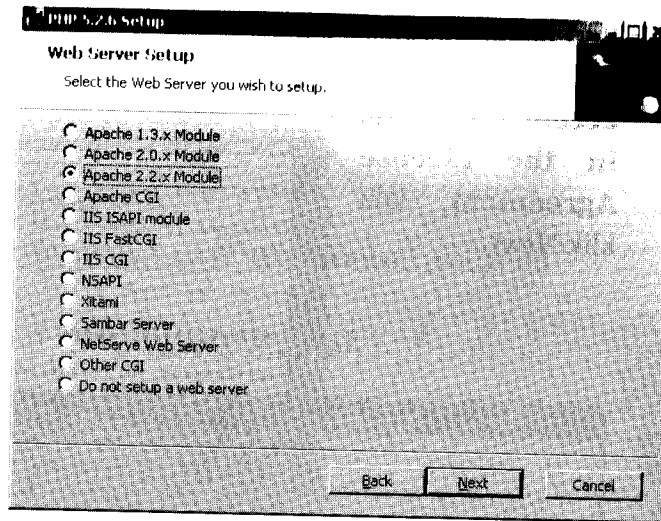
- ✓ Beri tanda *checklist* pada kotak **I accept the terms in the License Agreement**, lalu klik **Next**.



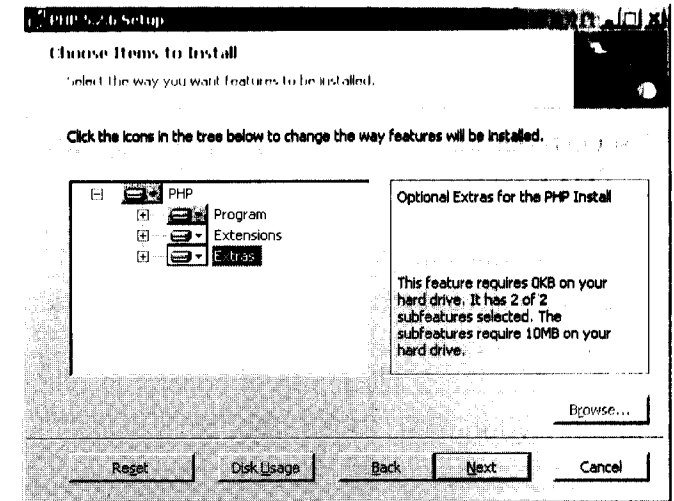
- ✓ Tentukan direktori tujuan instalasi. Sebagai latihan, isikan **C:\PHP**, lalu klik **Next**.



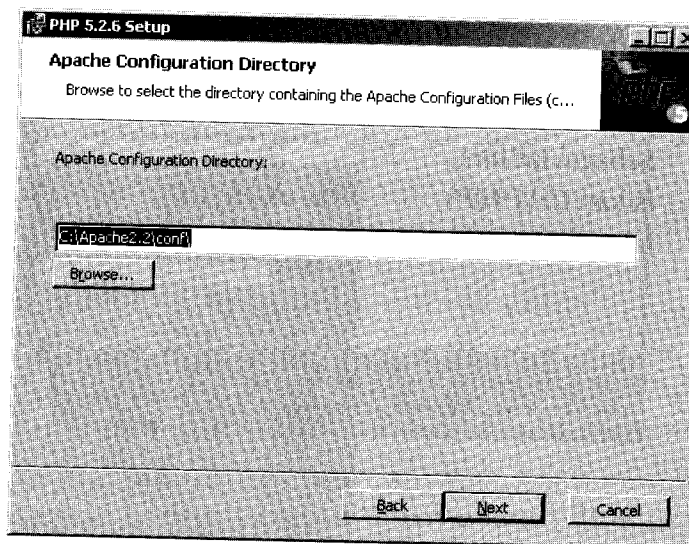
- ☑ Pilih **Apache 2.2.x Module**, lalu klik **Next**.



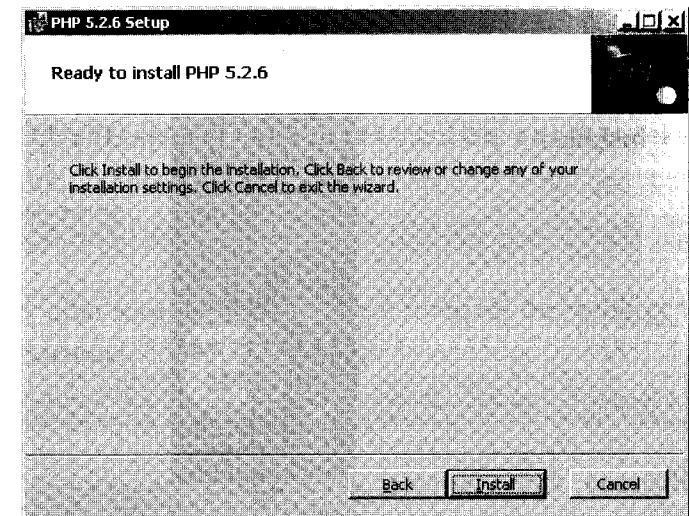
- ☑ Tentukan modul-modul PHP yang akan di-instal, lalu klik **Next**.



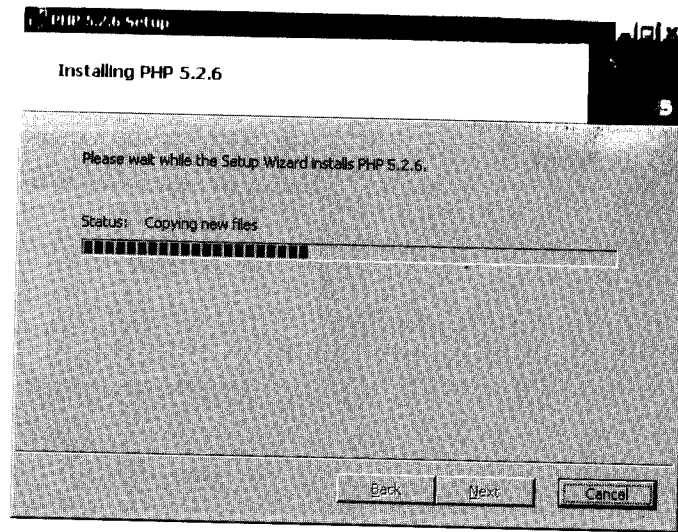
- ☑ Isikan **C:\Apache2.2\conf**. Direktori ini adalah tempat file **httpd.conf** berada. Klik **Next** untuk melanjutkan proses.



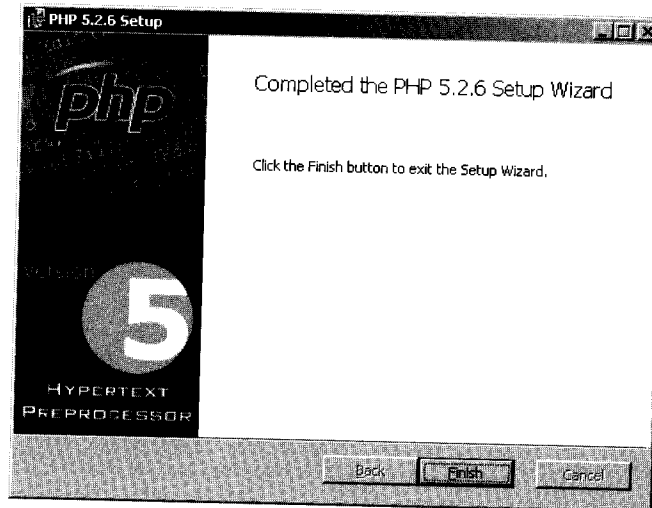
- ☑ Klik **Install** untuk memulai proses instalasi.



- ☑ Tunggu beberapa saat sampai muncul *form* seperti berikut:



- ☑ Klik **Finish** untuk mengakhiri proses instalasi.



Melakukan Konfigurasi PHP

Konfigurasi PHP disimpan di dalam *file* **php.ini**. Jika Anda menggunakan *software* PHP yang disertakan dalam CD dan mengikuti langkah-langkah instalasi di atas, maka *file* **php.ini** akan tersimpan di direktori **C:\PHP**.

Agar Anda dapat mempraktikkan praktikum-praktikum yang ada dalam modul ini, kita akan sedikit melakukan perubahan terhadap *file* **php.ini**.

Praktikum 2-2: Melakukan konfigurasi Apache

- ☑ Jalankan aplikasi Text Editor (usahakan yang mempunyai fasilitas nomor baris)
- ☑ Buka *file* **php.ini** yang ada di direktori **C:\PHP**.
- ☑ Pada baris 341, hilangkan tanda titik koma di depan teks, sehingga teks menjadi seperti berikut:

```
error_reporting = E_ALL & ~E_NOTICE
```

- ☑ Pada baris 372, ubah nilai **display_errors** menjadi **On**, sehingga teks menjadi seperti berikut:

```
display_errors = On
```

- ☑ Pada baris 490, ubah nilai **magic_quotes_gpc** menjadi **On**, sehingga teks menjadi seperti berikut:

```
magic_quotes_gpc = On
```

- ☑ Pada baris 529, isikan nilai untuk **doc_root**, sehingga teks menjadi seperti berikut:

```
doc_root = "c:\Apache2.2\htdocs"
```

- ☑ Simpan perubahan yang Anda lakukan pada *file* **php.ini**.
- ☑ *Copy file* **php5ts.dll** dari **C:\PHP** ke **C:\Apache2.2\bin**.

- ☑ Restart komputer Anda untuk memastikan semua konfigurasi telah terbaru.

Membuat Aplikasi “Hello World”

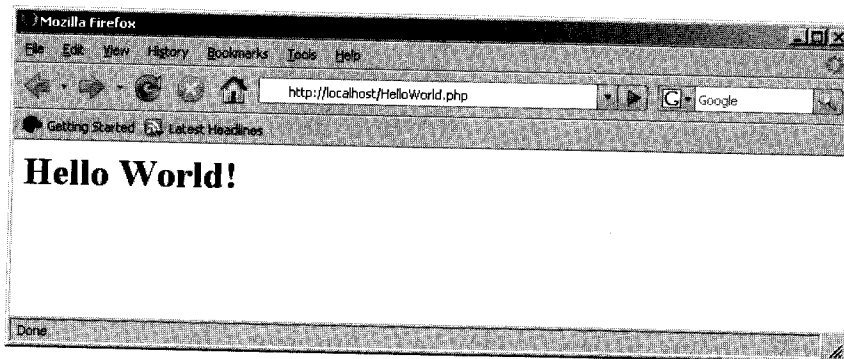
Untuk memeriksa apakah *software* PHP yang terpasang di komputer Anda sudah siap digunakan atau belum, Anda dapat membuat kode sederhana. Di sini, Anda akan diajak mempraktikkan cara pembuatan kode program PHP untuk pertama kalinya.

Praktikum 3-3: Membuat aplikasi “Hello World”

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode program berikut:

```
<?php
    echo "<h1>Hello World!</h1>";
?>
```

- ☑ Simpan *file* dengan nama **HelloWorld.php** di dalam direktori **C:\Apache2.2\htdocs**
- ☑ Jalankan *web browser* dan masukkan alamat berikut:
http://localhost/HelloWorld.php
- ☑ Jika semua berjalan mulus, maka Anda akan melihat tampilan seperti berikut:



Bagian 2

PHP Essential

Mengenai PHP

Setelah Anda memasang semua *software* yang diperlukan di komputer Anda, pada Modul 4 ini Anda akan mulai mempelajari tentang:

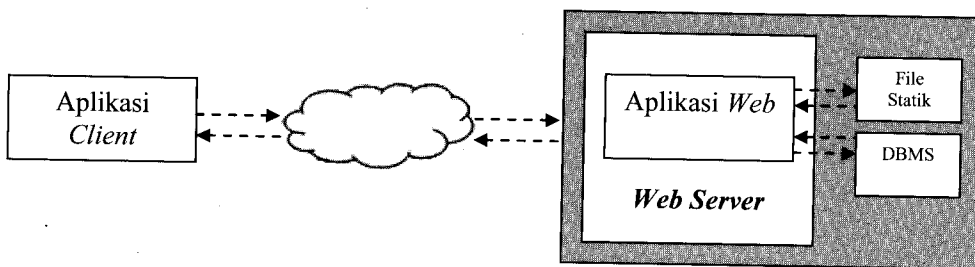
- Gambaran global tentang aplikasi *web*
- Cara kerja PHP
- Kerangka program PHP
- Memberikan komentar pada kode PHP
- Menggabungkan kode PHP dengan kode HTML

Catatan:

Pada saat mempelajari modul ini Anda dianggap telah mengetahui HTML. Bagi Anda yang belum tahu, Anda dapat mempelajarinya pada **Lampiran A – Referensi HTML**.

Gambaran Global Aplikasi Web

Aplikasi *web* adalah aplikasi yang disimpan dan dieksekusi di lingkungan *web server*. Setiap permintaan yang dilakukan oleh *user* melalui aplikasi klien (*web browser*) akan direspon oleh aplikasi *web* dan hasilnya akan dikembalikan lagi ke hadapan *user*. Dengan aplikasi *web*, halaman yang tampil di layar *web browser* dapat bersifat dinamis, tergantung dari nilai data atau parameter yang dimasukkan oleh *user*. Komunikasi antara *web browser* dan aplikasi *web* dapat digambarkan seperti berikut.



Seperti yang tampak pada gambar di atas bahwa aplikasi *web* dapat juga digunakan untuk mengakses *file-file* yang bersifat statis (misal: dokumen HTML, *file* gambar maupun *file* teks).

Berbeda dengan dokumen HTML dan kode pemrograman *client-side* lainnya (misal: JavaScript dan VBScript), kode dari aplikasi *web* tidak dapat dilihat atau dibaca oleh *user*.

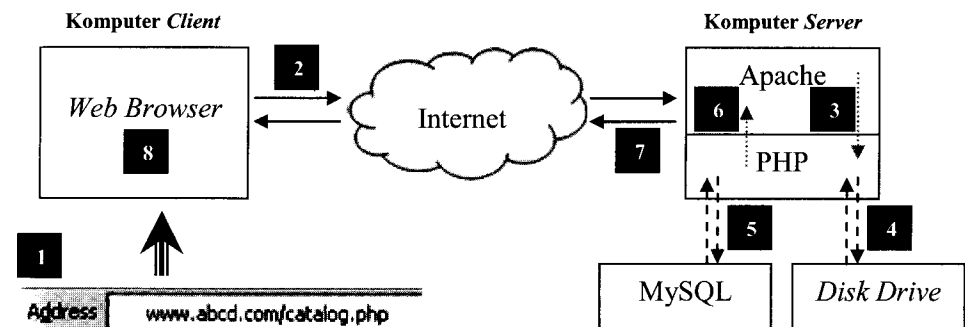
Sekilas tentang PHP

PHP adalah salah satu bahasa pemrograman skrip yang dirancang untuk membangun aplikasi *web*. Ketika dipanggil dari *web browser*, program yang ditulis dengan PHP akan di-*parse* di dalam *web server* oleh *interpreter* PHP dan diterjemahkan ke dalam dokumen HTML, yang selanjutnya akan ditampilkan kembali ke *web browser*. Karena pemrosesan program PHP dilakukan di lingkungan *web server*, PHP dikatakan sebagai bahasa sisi *server* (*server-side*). Oleh sebab itu, seperti yang telah dikemukakan sebelumnya, kode PHP tidak akan terlihat pada saat *user* memilih perintah “View Source” pada *web browser* yang mereka gunakan. Selain menggunakan PHP, aplikasi *web* juga dapat dibangun dengan Java (JSP – *JavaServer Pages* dan *Servlet*), Perl, maupun ASP (*Active Server Pages*).

Meskipun PHP 5 dapat digunakan untuk membuat aplikasi CLI (*Command Line Interface*) dan juga aplikasi desktop, namun pada umumnya orang menggunakan PHP untuk tujuan pembuatan aplikasi *web*.

Cara Kerja PHP

Cara kerja aplikasi *web* yang ditulis dengan PHP dapat diilustrasikan dengan gambar di bawah ini.



Berikut adalah keterangan dari gambar di atas:

1. *User* menulis *www.abcd.com/catalog.php* ke dalam *address bar* dari web browser (IE, Mozilla Firefox, Opera, dll)
2. *Web browser* mengirimkan pesan di atas ke komputer *server* (*www.abcd.com*) melalui internet, meminta halaman *catalog.php*
3. *Web server* (misalnya Apache), program yang berjalan di komputer *server*, akan menangkap pesan tersebut, lalu meminta interpreter PHP (program lain yang juga berjalan di komputer *server*) untuk mencari file *catalog.php* dalam *disk drive*.
4. *Interpreter* PHP membaca file *catalog.php* dari *disk drive*
5. *Interpreter* PHP akan menjalankan perintah-perintah atau kode PHP yang ada dalam file *catalog.php*. Jika kode dalam file *catalog.php* melibatkan akses terhadap *database* (misalnya MySQL) maka interpreter PHP juga akan berhubungan dengan MySQL untuk melaksanakan perintah-perintah yang berkaitan dengan *database*
6. *Interpreter* PHP mengirimkan halaman dalam bentuk HTML ke Apache
7. Melalui internet, Apache mengirimkan halaman yang diperoleh dari *interpreter* PHP ke komputer *user* sebagai respon atas permintaan yang diberikan
8. *Web browser* dalam komputer *user* akan menampilkan halaman yang dikirim oleh Apache

Sintaks Program PHP

PHP adalah bahasa yang dirancang untuk mudah diletakkan di dalam kode HTML. Banyak dijumpai kode PHP yang menyatu dengan kode HTML. Kode PHP diawali dengan tag **<?php** dan diakhiri dengan tag **?>**. Apabila kita melakukan konfigurasi terhadap file *php.ini* untuk mengizinkan penggunaan tag pendek (*short tag*) dengan mengubah nilai **short_open_tag** menjadi **On**, maka tag tersebut dapat diganti dengan **<?** dan **?>**. Dalam PHP 5, nilai *default* dari *short_open_tag* adalah *Off*. Selain

itu, PHP kita juga dapat menggunakan tag gaya ASP, **<%** dan **%>**, dengan mengubah nilai **asp_tags** dalam file *php.ini* menjadi **On**. Dalam modul ini, kita akan tetap menggunakan tag **<?php** dan **?>** untuk menulis kode-kode PHP.

Berikut ini contoh kode PHP yang sangat sederhana.

```
<?php
    echo "Hello World!";
?>
```

Perintah **echo** di dalam PHP berguna untuk mencetak nilai, baik teks maupun numerik, ke layar *web browser*. Selain **echo**, kita juga dapat menggunakan perintah **print**, sehingga kode di atas dapat juga ditulis seperti berikut.

```
<?php
    print "Hello World!";
?>
```

echo dan **print** juga memiliki bentuk penulisan yang lain, seperti layaknya pemanggilan fungsi. Berikut ini bentuk lain tersebut.

```
<?php
    echo("Hello World!");           // mencetak teks "Hello World!"
    echo(1);                        // mencetak angka 1
    print("Hello World!");          // mencetak teks "Hello World!"
    print(100);                     // mencetak angka 100
?>
```

Setiap perintah atau statemen di dalam kode PHP harus diakhiri dengan tanda titik koma atau *semicolon* (;).

Hasil yang akan diberikan oleh kedua kode di atas adalah sama. Meskipun demikian, penggunaan **echo** lebih banyak digemari oleh para *programmer web*.

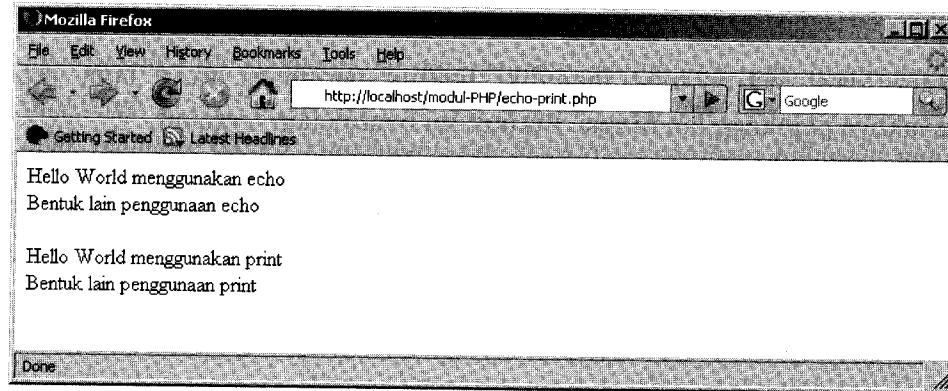
Praktikum 4-1: Menggunakan `echo` dan `print`

- ✓ Buatlah direktori kerja dengan nama modul-PHP di dalam `C:\Apache2.2\htdocs\`
- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<?php
    echo "Hello World\n";
    echo("Bentuk lain penggunaan echo\n\n");

    print "Hello World\n";
    print("Bentuk lain penggunaan print");
?>
```

- ✓ Simpan kode di atas ke dalam *file* dengan nama **echo-print.php** dan tempatkan di direktori `C:\Apache2.2\htdocs\modul-PHP`
- ✓ Jalankan *web browser* dan masukkan alamat:
`http://localhost/modul-PHP/echo-print.php`



Menggabungkan Kode PHP dengan Kode HTML

Kode PHP dapat juga disisipkan ke dalam kode HTML. Berikut ini contoh kode yang dapat menunjukkan hal tersebut.

Praktikum 4-2: Menyisipkan kode PHP dalam HTML

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

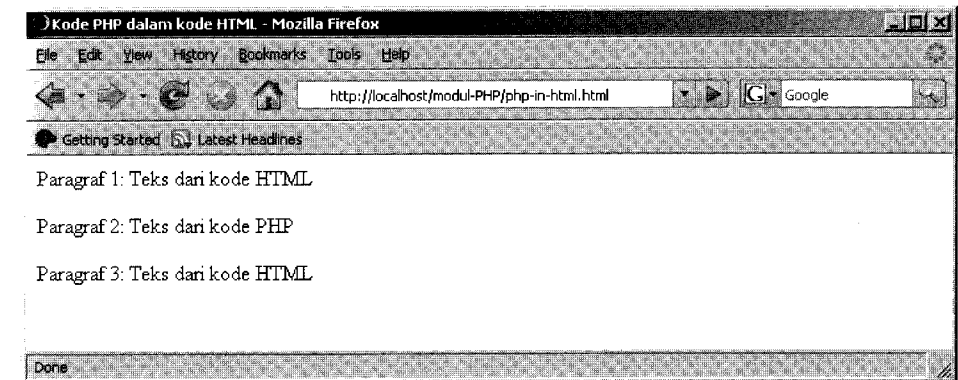
```
<html>
<head>
    <title>Kode PHP dalam kode HTML</title>
</head>
<body>
<p>Paragraf 1: Teks dari kode HTML</p>

<!-- Menyisipkan kode PHP dalam kode HTML -->
<?php
    echo "<p>Paragraf 2: Teks dari kode PHP</p>";
?>
<!-- akhir kode PHP -->

<p>Paragraf 3: Teks dari kode HTML</p>

</body>
</html>
```

- ✓ Simpan *file* dengan nama **php-in-html.html**
- ✓ Jalankan *web browser* dan masukkan alamat:
`http://localhost/modul-PHP/php-in-html.html`



Keterangan:

Pada kode di atas, teks pada paragraf pertama berasal dari kode HTML. Selanjutnya, kita menyisipkan kode PHP di bawahnya untuk menghasilkan teks pada paragraf kedua. Setelah kode PHP ditutup, kita juga masih membuat kode lagi untuk membuat teks pada paragraf ketiga.

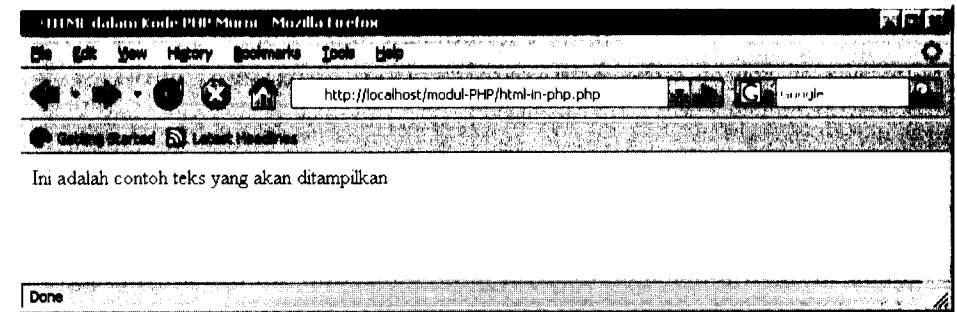
Ada pula programmer yang lebih suka membuat kode HTML dalam sintak PHP murni. Meskipun demikian, penulis menyarankan agar Anda tidak menggunakan gaya penulisan seperti ini karena berdasarkan pengalaman gaya kode semacam ini lebih susah untuk dilacak jika di dalam kode tersebut ternyata terdapat kesalahan (kode lebih sulit untuk di-debug). Berikut ini contoh sederhana yang menunjukkan kode HTML dapat ditulis dalam kode PHP murni.

Praktikum 4-3: Menulis HTML dalam kode PHP

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<?php
echo "<html>";
echo "<head>";
echo "<title>HTML dalam Kode PHP Murni</title>";
echo "</head>";
echo "<body>";
echo "Ini adalah contoh teks yang akan ditampilkan";
echo "</body>";
echo "</html>";
?>
```

- ☒ Simpan file dengan nama **html-in-php.php**
- ☒ Jalankan web browser dan masukkan alamat:
http://localhost/modul-PHP/html-in-php.php



Keterangan:

Pada kode di atas semua tag HTML yang ada ditulis dalam bentuk string, yang kemudian ditampilkan menggunakan perintah `echo`.

Memberikan Komentar pada Kode PHP

Komentar berfungsi untuk memberikan informasi kepada orang yang membaca kode program yang kita buat, tapi baris-baris komentar tersebut akan diabaikan oleh PHP (tidak ikut terkompilasi sebagai kode program).

Contoh:

```
// 0 = Lajang, 1 = Menikah, 2 = Duda, 3 = Janda
$status = 0;
```

PHP menyediakan beberapa cara untuk memasukkan komentar ke dalam baris kode program, yang semuanya diadopsi dari bahasa yang telah ada sebelumnya seperti C/C++.

Gaya Bahasa C

Jenis komentar milik bahasa C ini bisa berlaku untuk komentar yang berjumlah lebih dari satu baris. PHP juga memberlakukan jenis komentar seperti ini.

Contoh:

```
/* Ini adalah contoh komentar  
yang dibuat menggunakan  
kode gaya bahasa C */
```

Gaya Bahasa C++

Berbeda dengan jenis sebelumnya, jenis komentar ini dibuat menggunakan *double slash* (//) dan hanya berlaku untuk satu baris.

Contoh:

```
// Ini adalah komentar PHP yang diadopsi dari bahasa C++
```

Gaya Unix Shell

Terlepas dari apa alasannya, jenis komentar ini jarang digunakan oleh para *programmer* PHP. Komentar jenis ini dibuat dengan cara membubuhkan tanda # di depannya.

Contoh:

```
#####  
# Ini adalah komentar gaya Unix Shell  
#####
```

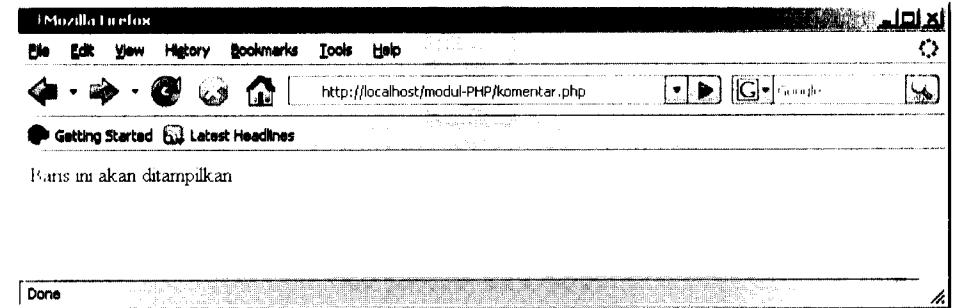
Praktikum 4-4: Menulis komentar dalam kode PHP

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<?php  
    // echo "Baris ini tidak akan ditampilkan";  
    echo "Baris ini akan ditampilkan";  
    /* echo "Baris ini juga tidak akan ditampilkan"; */  
    # echo "dan yang ini juga tidak akan ditampilkan";  
?>
```

- ☒ Simpan *file* dengan nama **komentar.php**

- ☒ Jalankan *web browser* dan masukkan alamat:
http://localhost/modul-PHP/komentar.php



Keterangan:

Anda bisa lihat bahwa yang akan dianggap sebagai perintah program hanyalah baris ke-2. Baris ke-1, ke-3, dan ke-4 tidak akan ikut dieksekusi karena merupakan komentar program. Pada baris ke-1, komentar dibuat menggunakan tanda //; baris ke-2, komentar dibuat menggunakan tanda /* dan */; sedangkan baris ke-3 menggunakan tanda #.

Tipe Data, Variabel, dan Operator

Pada modul sebelumnya, Anda sudah mempelajari tentang konsep dasar aplikasi *web* yang ditulis menggunakan PHP serta langkah-langkah pembuatan program PHP yang dipadu dengan kode HTML. Dalam modul ini kita akan melanjutkan pembahasan dengan materi-materi berikut:

- Pengelompokan tipe data dalam PHP
- Pengenalan dan penggunaan variabel
- Pengenalan dan penggunaan operator

Pengelompokan Tipe Data

PHP mendukung delapan tipe nilai atau tipe data. Empat nilai skalar (*single-value*): *integer*, *floating-point*, *string*, dan *boolean*; dua nilai koleksi: *array* dan objek. Sisanya adalah tipe khusus: *resource* dan *NULL*.

Catatan:

Sebelum Anda mempelajari tentang penggunaan variabel, Anda harus memahami tipe data terlebih dahulu.

Integer

Integer adalah tipe data yang merepresentasikan bilangan utuh atau bulat, seperti 12, 14, 2009, dan sebagainya. Rentang yang diperbolehkan adalah dari -2,147,483,648 sampai +2,147,483,647.

Nilai literal dari tipe *integer* dapat ditulis dalam bentuk desimal (basis 10), oktal (basis 8), maupun heksadesimal (basis 16).

Contoh *integer* yang ditulis dalam bentuk desimal:

```
2009
-212
+13
```

Contoh *integer* yang ditulis dalam bentuk oktal:

```
0755 // desimal 493
+010 // desimal 8
```

Contoh *integer* yang ditulis dalam bentuk heksadesimal:

```
0xFF // desimal 255
0x10 // desimal 10
-0xDAD1 // desimal -56017
```

Floating-point

Floating-point number (sering juga dinamakan dengan bilangan *real*) adalah tipe data yang merepresentasikan nilai-nilai numerik dalam bentuk pecahan atau mengandung angka desimal di belakang koma. *Floating-point* dalam PHP selaras dengan tipe *double* dalam bahasa C, biasanya berada dalam rentang 1.7E-308 dan 1.7E+308 dengan 15 digit akurasi.

Bilangan *floating-point* dapat ditulis dalam notasi normal dan notasi sains (*scientific*).

Contoh:

```
3.14
0.017
7.1
0.314E1 // 0.314*101, or 3.14
17.0E-3 // 17.0*10-3, or 0.017
```

Floating-point sering merupakan representasi “kira-kira” dari suatu bilangan. Sebagai contoh, 3.5 secara aktual direpresentasikan dengan nilai 3.4999999999999.

String

PHP mendukung dua tipe *string* yaitu yang diapit oleh petik ganda dan tunggal. Dalam pembuatan *web*, *string* merupakan tipe data yang kerap kali dijumpai dalam berbagai kasus.

Contoh:

```
'string dalam petik tunggal'
"string dalam petik ganda"
```

String yang diapit oleh petik ganda juga dapat mendukung *escape sequence*, yaitu beberapa karakter yang dianggap sebagai satu karakter, yang digunakan untuk merepresentasikan kedudukan karakter tertentu.

Escape sequence	Karakter yang direpresentasikan
\"	Double quotes
\n	Newline
\r	Carriage Return
\t	Tab
\\	Backslash
\\$	Dollar sign
\{	Left brace
\}	Right brace
\[	Left bracket
\]	Right bracket
\0 sampai \777	Karakter ASCII yang direpresentasikan dengan nilai oktal
\x0 sampai \xFF	Karakter ASCII yang direpresentasikan dengan nilai heksadesimal

Boolean

Tipe *boolean* digunakan untuk merepresentasikan nilai kebenaran (*truth value*). Nilai yang mungkin dimasukkan ke dalam variabel *boolean* adalah *true* dan *false*. Tipe ini biasanya dijumpai pada pendefinisian kondisi, baik dalam struktur pemilihan maupun pengulangan.

```
// Contoh penggunaan boolean dalam struktur pemilihan
if ($ketemu) {
    // lakukan sesuatu
}

// Contoh penggunaan boolean dalam
for ($i=0; $i < 10; $i++) {
    // lakukan sesuatu
}
```

Dalam PHP, berikut ini adalah nilai-nilai *false*:

- *Keyword false*
- *Integer 0*
- *Floating-point 0.0*
- *String* kosong ("") dan *string "0"*
- *Array* dengan nol elemen
- Objek yang tidak memiliki nilai atau fungsi
- Nilai *NULL*

Array

Array adalah variabel yang menyimpan sekelompok nilai, yang dapat diidentifikasi berdasarkan posisinya (nomor, dengan 0 sebagai indeks pertamanya) atau dengan nama (bertipe *string*) yang telah didefinisikan sebelumnya.

```
$person[ 0] = "Aas";
$person[ 1] = "Jajang";
$person[ 3] = "Dadan";
$person[ 4] = "Renra"

$pegawai[ 'manajer' ] = "Aas";
$pegawai[ 'dirut' ] = "Jajang";
$pegawai[ 'umum' ] = "Dadan";
$pegawai[ 'teknisi' ] = "Renra";
```

Berikut ini konstruksi untuk membentuk suatu *array*:

```
$person = array('Aas', 'Jajang', 'Dadan', 'Renra');
$pegawai = array('manajer' => 'Aas',
                  'dirut'   => 'Jajang',
                  'umum'    => 'Dadan',
                  'teknisi' => 'Renra');
```

Terdapat beberapa cara untuk mengakses nilai *array* melalui blok pengulangan, namun yang paling lazim adalah dengan menggunakan struktur pengulangan *foreach*.

```
foreach($person as $name) {
    echo "Hallo, $name\n";
}
```

```
foreach($pegawai as $jabatan => $pejabat) {
    echo "$jabatan dijabat oleh $pejabat";
}
```

Objek

PHP mendukung *Object-Oriented Programming* (OOP), yang dapat memudahkan dalam proses *debugging*, *maintenance*, dan penggunaan ulang kode (*reuseable code*). Dalam OOP, kelas merupakan unit terpenting. Kelas adalah suatu definisi dari struktur yang memiliki properti dan metode (fungsi). Dalam PHP, kelas didefinisikan menggunakan kata kunci **class**.

Contoh:

```
class Person {
    var $nama = "";
    function nama($newname = NULL) {
        if (!is_null($newname)) {
            $this->nama = $newname;
        }
        return $this->nama;
    }
}
```

Sekali kelas didefinisikan, Anda dapat membentuk objek dengan menggunakan kata kunci **new**. Properti dan metode dapat diakses melalui operator **->**.

```
$person1 = new Person;
$person1->nama('Aas Asyanto');
printf("Hallo %s, apa kabar?", $person1->nama);
```

Anda dapat memeriksa suatu nilai atau variabel apakah berbentuk objek atau bukan dengan menggunakan fungsi `is_object()`.

```
if (is_object($obj)) {
    // $obj adalah sebuah objek
}
```

Pengenalan Variabel dan Cara Pembuatannya

Variabel adalah suatu pengenal di dalam program yang merepresentasikan nilai atau data tertentu. Dengan memasukkan nilai ke dalam variabel, maka Anda dapat mengacu variabel tersebut dari tempat lain dalam kode program yang Anda buat.

Dalam PHP, variabel didefinisikan dengan cara membubuhkan tanda **\$** (*dollar*) di depannya. Berikut ini langkah-langkah yang diperlukan untuk membuat variabel di dalam PHP:

1. Tulislah nama variabel yang Anda butuhkan:

```
username
usia
```

2. Tempatkan tanda **\$** di depan nama variabel tersebut:

```
$username
$usia
```

3. Gunakan tanda sama dengan (=) setelah nama untuk memasukkan nilai literal ke dalam variabel tersebut:

```
$username = "pras"
$usia = 27
```

4. Memasukkan nilai ke dalam variabel merupakan sebuah perintah, maka Anda harus mengakhirinya dengan tanda titik koma atau *semicolon* (;):

```
$username = "pras";
$usia = 27;
```

Berbeda dengan bahasa pemrograman lain pada umumnya, dalam PHP, variabel tidak perlu dideklarasikan terlebih dahulu. Selain itu, variabel PHP dapat menyimpan nilai dari tipe data apa pun. Perhatikan contoh kode berikut:

```
$x = "Dona";  
$x = 12;  
$x = 4.23;  
$x = false;  
$x = array("Dona", "12", "4.23", "false");  
  
$detik_per_hari = 60 * 60 * 24;  
echo "Dalam 1 hari terdapat $detik_per_hari detik.";
```

Seperti yang Anda lihat dalam kode di atas:

- Pada baris pertama variabel `$x` menyimpan nilai *string*
- Pada baris kedua variabel `$x` menyimpan nilai *integer*
- Pada baris ketiga variabel `$x` menyimpan nilai *floating-point*
- Pada baris keempat variabel `$x` menyimpan nilai *boolean*
- Pada baris kelima variabel `$x` menyimpan nilai *array*

Praktikum 5-1: Membuat dan Menggunakan Variabel

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>  
<head>  
  <title>Variabel</title>  
</head>  
<body>  
<h2>Demo membuat dan menggunakan variabel</h2>  
<?php  
  
// membuat variabel dan mengisikan nilai ke dalamnya  
$varString = "variabel bertipe string";  
$varFloat = 3.14;  
$varInteger = 212;
```

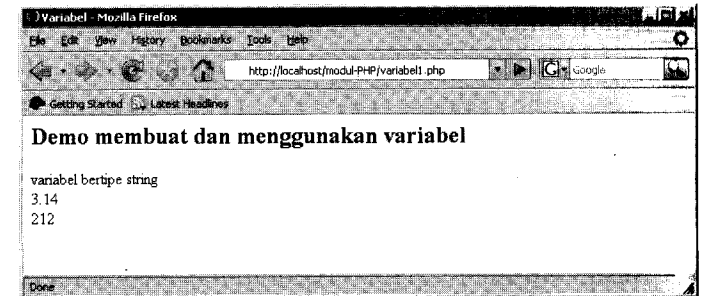
```
// mencetak nilai variabel ke layar browser  
echo $varString . "<br />";  
echo $varFloat . "<br />";  
echo $varInteger;  
?  
  
</body>  
</html>
```

- ✓ Simpan file dengan nama **variabel1.php**
- ✓ Jalankan web browser dan masukkan **http://localhost/modul-PHP/variabel1.php**

Keterangan:

Seperti yang tampak pada kode di atas, ketika Anda ingin memasukkan nilai bertipe string ke

dalam suatu variabel, Anda harus mengapit nilai tersebut dengan tanda petik ganda (""); sedangkan untuk data dengan tipe floating-point (bilangan riil) dan integer (bilangan bulat) Anda tidak perlu menggunakan tanda petik.



Dalam PHP, tanda titik (.) digunakan untuk menyambung teks atau string. Adapun `<br/>` yang terdapat pada kode di atas adalah tag HTML yang berguna untuk ganti baris.

Penamaan Variabel

Meskipun terlihat sepele, namun sebenarnya penamaan variabel di dalam PHP perlu mendapat perhatian khusus. Berikut ini aturan-aturan penamaan yang berlaku:

- Nama variabel tidak boleh mengandung spasi

Contoh:

```
$my var
$orang tua wali
```

- Nama variabel tidak boleh diawali dengan karakter angka

Contoh:

```
$2D
$3Dimensi
```

- Selain tanda \$ yang terdapat di depan nama, nama variabel tidak boleh menggunakan karakter simbol, kecuali tanda _ (*underscore*)

Contoh:

```
$bilangan?
$tag<br>
```

- Nama variabel dalam PHP bersifat *case-sensitive* sehingga \$YOURVAR dan \$yourvar akan dianggap sebagai dua variabel yang berbeda
- Agar kode program mudah dibaca oleh orang lain, usahakan untuk menggunakan nama variabel yang mudah dimengerti maksudnya. Sebagai contoh, jangan gunakan \$n untuk merepresentasikan nama dan \$p untuk merepresentasikan *password*. Jika Anda menggunakan nama \$nama dan \$password, itu akan lebih baik.

Variabel Variabel

Kita dapat mengacu nilai dari variabel yang namanya tersimpan dalam variabel lain. Sebagai contoh:

```
$var1 = "var2";
$$var1 = "Dorrr!!!";
```

Setelah statemen kedua dieksekusi, \$var1 akan bernilai "Dorrr!!!".

Referensi Variabel

Referensi merupakan alias dari suatu variabel. Untuk membuat \$hitam sebagai alias dari variabel \$putih, diperlukan kode seperti berikut:

```
$hitam =& $putih;
```

Melalui kode tersebut, nilai lama dari variabel \$hitam akan hilang, dan akan tertimpa oleh nilai yang disimpan dalam variabel \$putih. Hal ini bisa terjadi karena variabel \$hitam sekarang adalah nama lain atau alias dari variabel \$putih.

Variabel Global dan Lokal

Variabel yang dideklarasikan di dalam fungsi akan bersifat lokal untuk fungsi tersebut. Dengan demikian, variabel tersebut hanya akan dapat diakses oleh proses-proses yang terdapat dalam fungsi tersebut, termasuk *nested-function* atau sub-fungsi. Sedangkan variabel yang berada di luar fungsi secara *default* akan dianggap sebagai variabel global.

```
function update_counter() {
    $counter++;
}

$counter = 10;
update_counter();
echo $counter;
```

Perhatikan perbedaan hasil dengan kode berikut:

```
function update_counter() {
    global $counter;
    $counter++;
}

$counter = 10;
update_counter();
echo $counter;
```


Variabel Statis

Variabel statis adalah variabel yang dapat menyimpan nilai terakhir dari setiap pemanggilan fungsi. Perhatikan contoh berikut:

```
function update_counter() {  
    static $counter = 0;  
    $counter++;  
    echo "Nilai \${counter} di dalam fungsi: \${counter}\n";  
}  
  
$counter = 10;  
update_counter();  
update_counter();  
echo "\${counter} global adalah: \${counter}\n";
```

Parameter Fungsi

Variabel dapat berperan sebagai parameter suatu fungsi. Caranya pendefinisianya adalah sebagai berikut:

```
function SelamatDatang($nama) {  
    echo "Halo $name, selamat datang di web site kami.\n";  
}  
  
SelamatDatang("Herlan Andini Setiawan");
```

Fungsi akan dikaji lebih jauh pada bagian lain.

Konstanta

Konstanta adalah suatu pengenalan program yang digunakan untuk menyimpan nilai sederhana, sama halnya seperti variabel. Perbedaannya, sekali didefinisikan, nilai konstanta tidak dapat diubah lagi; sedangkan nilai dari variabel dapat diubah kapan saja sesuai dengan kebutuhan. Pendefinisian nilai konstanta dilakukan dengan menggunakan fungsi `define()`.

Contoh:

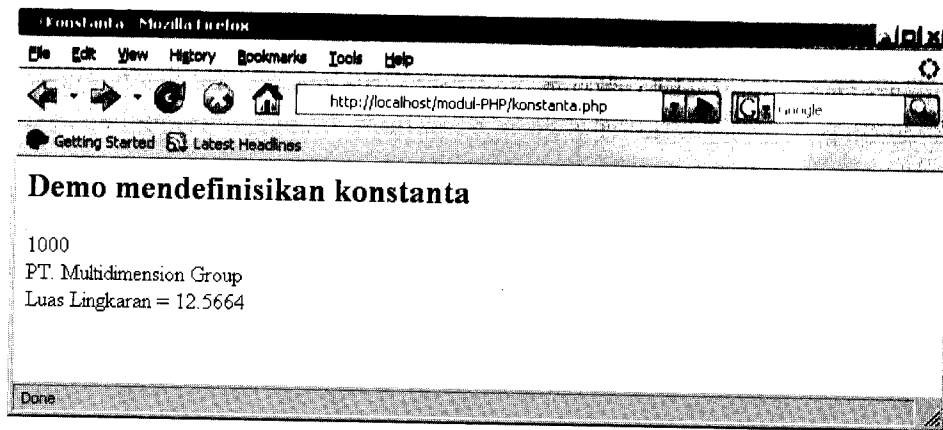
```
define('NAMA_PERUSAHAAN', "PT. Multidimension Group");  
echo NAMA_PERUSAHAAN;  
  
define('PI', 3.1416);  
$luas = PI * $r * $r;
```

Praktikum 5-2: Mendefinisikan dan menggunakan konstanta

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>  
<head>  
    <title>Konstanta</title>  
</head>  
<body>  
<h2>Demo mendefinisikan konstanta</h2>  
  
<?php  
    // mendefinisikan konstanta  
    define('MAKSIMAL', 1000);  
    define('NAMA_PERUSAHAAN', "PT. Multidimension Group");  
    define('PI', 3.1416);  
  
    // mencetak nilai konstanta ke layar browser  
    echo MAKSIMAL . "<br />";  
    echo NAMA_PERUSAHAAN . "<br />";  
  
    // menggunakan konstanta dalam perhitungan  
    $jarijari = 2;  
    $luasLingkaran = PI * $jarijari * $jarijari;  
    echo "Luas Lingkaran = " . $luasLingkaran;  
?>  
  
</body>  
</html>
```

- ☒ Simpan file dengan nama **konstanta.php**
- ☒ Jalankan web browser dan masukkan **http://localhost/modul-PHP/konstanta.php**



Keterangan:

Pada praktikum kali ini, kita mendefinisikan tiga buah konstanta dengan nama *MAKSIMAL*, *NAMA_PERUSAHAAN*, dan *PI*. Hal yang perlu Anda perhatikan adalah pada saat pemanggilan konstanta. Pemanggilan konstanta tidak boleh menggunakan tanda \$ di depannya (berbeda dengan pemanggilan variabel).

Operator dalam PHP

Operator dalam PHP dapat dikategorikan menjadi beberapa kelompok, yaitu:

- operator *assignment* (pengisian)
- aritmetika
- *relational*
- logika

Operator Assignment

Operator *assignment* adalah operator yang digunakan untuk mengisi nilai ke dalam suatu variabel. PHP menggunakan operator sama dengan (=)

untuk melakukan hal ini, seperti layaknya bahasa C/C++ dan Java. Sebagai bahan perbandingan, dalam bahasa Pascal/Delphi operator yang digunakan sebagai operator *assignment* adalah *:=*. Berikut ini contoh beberapa baris kode yang menunjukkan proses pengisian nilai di dalam PHP.

```
1 <?php
2     $b11 = 78;
3     $b12 = 42.39;
4     $ch = 'B';
5     $nama = "Joe";
6     // ...
7
```

Operator Aritmetika

PHP memiliki dukungan terhadap operasi perhitungan standar dengan menyediakan operator-operator aritmetika. Berikut ini daftar operator yang termasuk ke dalam kelompok ini.

Operator	Keterangan
+	Operator untuk penjumlahan
-	Operator untuk pengurangan
*	Operator untuk perkalian
/	Operator untuk pembagian
%	Operator untuk mengetahui sisa dari pembagian (modulus)
++	Operator untuk menaikkan nilai variabel sebesar 1 (<i>increment</i>)
--	Operator untuk menurunkan nilai variabel sebesar 1 (<i>decrement</i>)

Praktikum 5-3: Menggunakan operator aritmetika

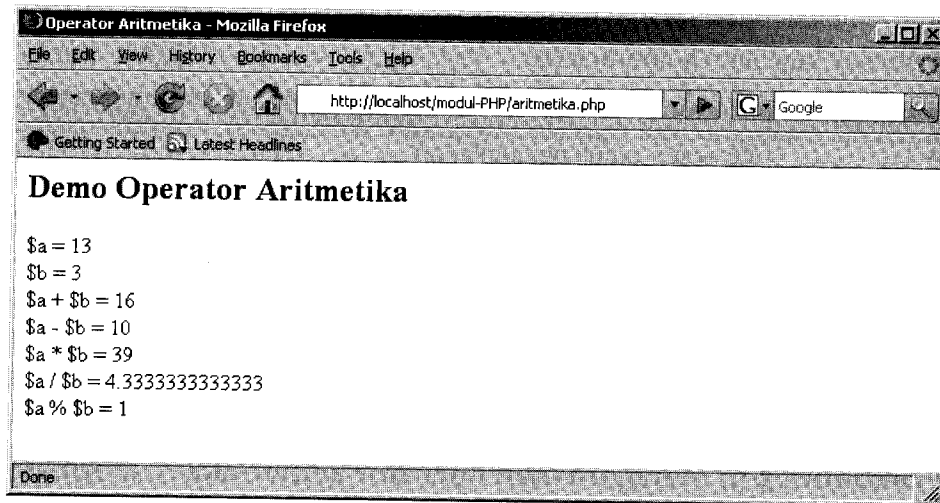
- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
    <title>Operator Aritmetika</title>
</head>
<body>
```

```
<h2>Demo Operator Aritmetika</h2>
```

```
<?php
    $a = 13;
    $b = 3;
    echo "\$a = " . $a;
    echo "<br />\$b = " . $b;
    echo "<br />\$a + \$b = " . ($a + $b);
    echo "<br />\$a - \$b = " . ($a - $b);
    echo "<br />\$a * \$b = " . ($a * $b);
    echo "<br />\$a / \$b = " . ($a / $b);
    echo "<br />\$a % \$b = " . ($a % $b);
?>
</body>
</html>
```

- ☑ Simpan file dengan nama **aritmetika.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/aritmetika.php**



Keterangan:

Perlu Anda perhatikan penggunaan operator % di atas. Operator ini akan menghasilkan sisa bagi dari operasi pembagian yang dilakukan. Dalam contoh ini, kita melakukan pembagian antara bilangan 13 dan 3. 13

dibagi 3 menghasilkan nilai 4 dan sisa baginya adalah 1. Dengan demikian 13 % 3 menghasilkan nilai 1.

Operator ++ (increment) dan -- (decrement) merupakan operator unary, yaitu operator yang hanya memiliki satu buah operand. Operator increment dibedakan menjadi dua: pre-increment dan post-increment. Begitu juga dengan operator decrement, dibedakan menjadi pre-decrement dan post-decrement.

Pada pre-increment, nilai variabel akan dinaikkan terlebih dahulu sebelum diproses; sedangkan pada post-increment, nilai variabel diproses terlebih dahulu sebelum dinaikkan.

Praktikum 5-4: Menggunakan operator increment dan decrement

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>Operator Increment dan Decrement</title>
</head>
<body>
<h2>Demo operator ++ dan --</h2>
<?php

    $x = 10;

    echo "<p>";
    echo "Pre-increment";
    echo "<br />Nilai \$x awal = \$x";
    echo "<br />Nilai ++\$x = " . ++$x;
    echo "<br />Nilai \$x akhir = \$x";
    echo "</p>";

    $x = 10; // mengubah kembali nilai x menjadi 10

    echo "<p>";
    echo "Post-increment";
```

```

echo "<br />Nilai \$x awal = \$x";
echo "<br />Nilai \$x++ = " . \$x++;
echo "<br />Nilai \$x akhir = \$x";
echo "</p>";

```

```

$y = 20;

```

```

echo "<p>";
echo "Pre-decrement";
echo "<br />Nilai \$y awal = \$y";
echo "<br />Nilai --\$y = " . --\$y;
echo "<br />Nilai \$y akhir = \$y";
echo "</p>";

```

```

$y = 20; // mengubah kembali nilai y menjadi 20

```

```

echo "<p>";
echo "Post-decrement";
echo "<br />Nilai \$y awal = \$y";
echo "<br />Nilai \$y-- = " . \$y--;
echo "<br />Nilai \$y akhir = \$y";
echo "</p>";

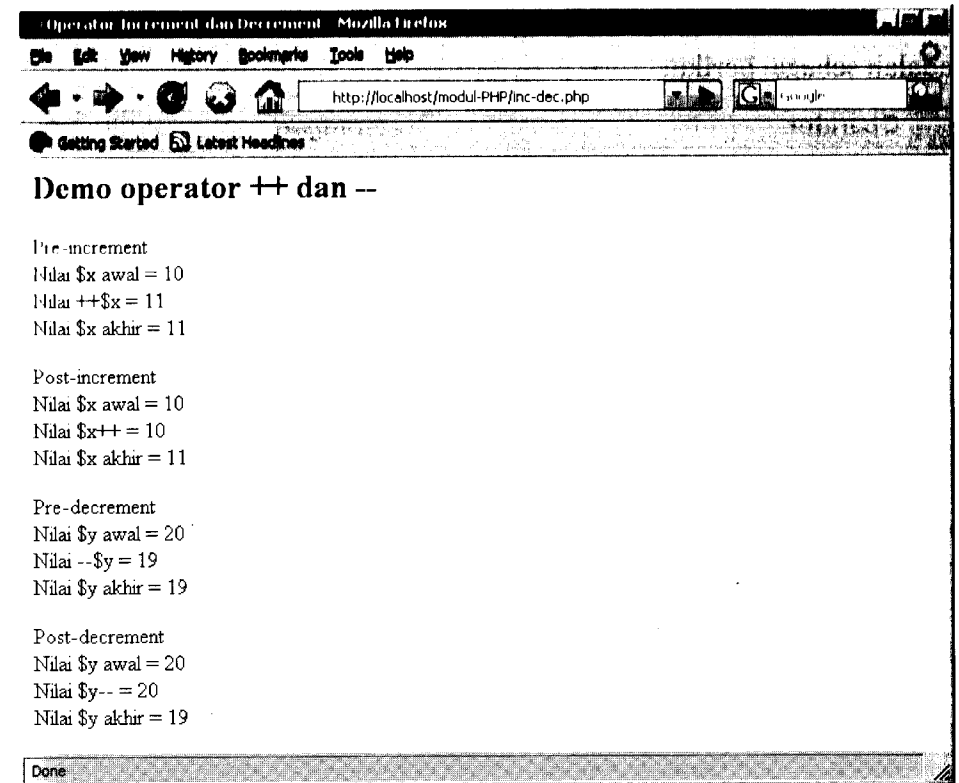
```

```

?>
</body>
</html>

```

- ☑ Simpan file dengan nama **inc-dec.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/inc-dec.php**



Keterangan:

Pada pre-increment nilai $\$x$ akan dinaikkan dahulu sebelum ditampilkan sehingga ketika ditampilkan nilai yang muncul adalah 11, sedangkan pada post-increment nilainya ditampilkan terlebih dahulu sebelum akhirnya dinaikkan. Dengan demikian, nilai yang muncul masih tetap 10. Untuk pre-decrement dan post-decrement, konsepnya sama seperti increment.

Operator Relasional

Operator relasional digunakan untuk membandingkan dua buah nilai. Hasil dari operasi perbandingan ini selalu menghasilkan nilai *boolean* (true

atau false). Dalam PHP, yang termasuk ke dalam operator relasional adalah sebagai berikut.

Operator	Keterangan
\$a == \$b	Memeriksa apakah \$a sama dengan \$b
\$a != \$b	Memeriksa apakah \$a tidak sama dengan \$b
\$a < \$b	Memeriksa apakah \$a lebih kecil dari \$b
\$a > \$b	Memeriksa apakah \$a lebih besar dari \$b
\$a <= \$b	Memeriksa apakah \$a lebih kecil atau sama dengan \$b
\$a >= \$b	Memeriksa apakah \$a lebih besar atau sama dengan \$b

Operasi perbandingan seperti ini banyak digunakan sebagai kondisi dalam blok pemilihan (if-else) dan pengulangan (for, while, dan do-while). Untuk memahami cara kerja masing-masing operator di atas, silakan lakukan praktikum berikut ini.

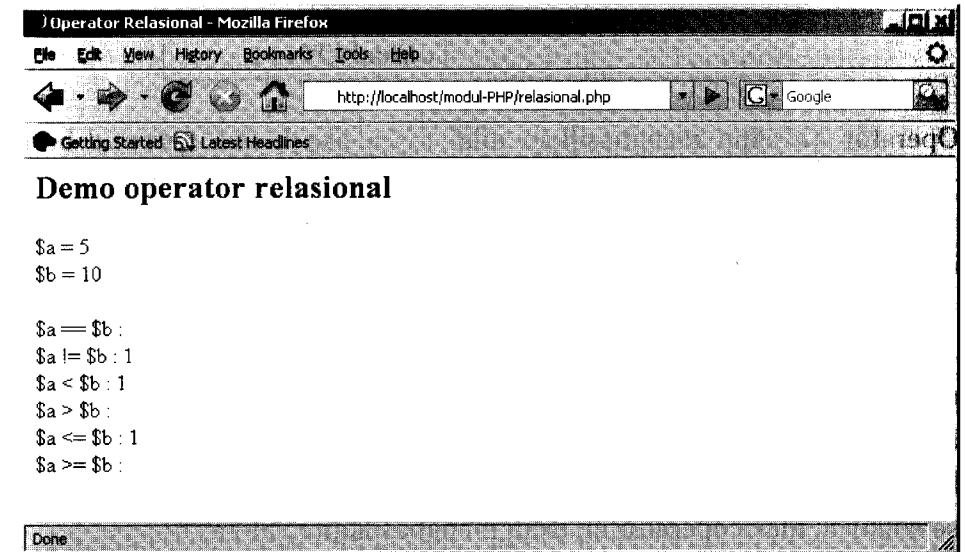
Praktikum 5-5: Menggunakan operator relasional

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Operator Relasional</title>
</head>
<body>
  <h2>Demo operator relasional</h2>
  <?php
    $a = 5;
    $b = 10;
    echo "\$a = $a";
    echo "<br />\$b = $b";
    echo "<br />";
    echo "<br />\$a == \$b : " . ($a == $b);
    echo "<br />\$a != \$b : " . ($a != $b);
    echo "<br />\$a < \$b : " . ($a < $b);
```

```
echo "<br />\$a > \$b : " . ($a > $b);
echo "<br />\$a <= \$b : " . ($a <= $b);
echo "<br />\$a >= \$b : " . ($a >= $b);
?>
</body>
</html>
```

- ✓ Simpan file dengan nama **relasional.php**
- ✓ Jalankan web browser dan masukkan **http://localhost/modul-PHP/relasional.php**



Keterangan:

Pada praktikum ini kita memasukkan nilai 5 ke dalam variabel \$a dan 10 ke dalam variabel \$b. Selanjutnya, kita merelasikan kedua variabel tersebut menggunakan operator relasional. Hasilnya adalah seperti yang tampak pada gambar di atas. Nilai 1 yang ditampilkan pada hasil di atas merepresentasikan nilai **true**.

Operator Logika

Operator logika digunakan untuk mengoperasikan dua buah *operand* yang bertipe *boolean*. Hasil yang diberikan dari operasi ini juga akan bertipe *boolean*. Operasi logika dapat berupa AND (dan), OR (atau), dan NOT (negasi). Tabel di bawah ini menunjukkan operator logika dalam PHP.

Operator	Keterangan
&&	Logika AND
	Logika OR
!	Logika NOT

Operator && akan menghasilkan nilai *true* jika kedua *operand* bernilai *true*. Selain kondisi itu, operator ini akan menghasilkan nilai *false*.

Operator || akan menghasilkan nilai *false* jika kedua *operand* bernilai *false*. Selain kondisi tersebut, operator || akan menghasilkan nilai *true*.

Operator ! digunakan untuk melakukan negasi terhadap suatu nilai *boolean*. Operator ini hanya memiliki sebuah *operand*. Jika *operand* bernilai *true*, maka operator ini akan menghasilkan nilai *false*. Jika sebaliknya, maka operator ini akan menghasilkan nilai *false*.

Mengontrol Alur Program

Dalam Modul 6 ini Anda akan mempelajari tentang tata cara melakukan pengontrolan jalannya alur program. Pembahasan akan diklasifikasikan menjadi tiga bagian, yaitu:

- Teknik dan tata cara melakukan pemilihan statemen
- Teknik dan tata cara mengulang statemen

Pemilihan Statemen

Blok pemilihan yang terdapat pada suatu program berguna untuk menentukan statemen mana yang akan dieksekusi, yang didasarkan pada

kondisi-kondisi tertentu yang didefinisikan. Jika kondisi terpenuhi (bernilai `true`) maka statemen yang terdapat di dalam blok bersangkutan akan dieksekusi. Sebaliknya, jika tidak terpenuhi (bernilai `false`) maka statemen tersebut akan diabaikan oleh program. Dalam PHP, pemilihan statemen dapat dilakukan dengan dua cara, yaitu:

- Menggunakan statemen `if`
- Menggunakan statemen `switch`

Menggunakan Statemen `if`

Statemen `if` memiliki bentuk umum penggunaan seperti berikut:

`if (kondisi) statemen`

Jika statemen yang akan dieksekusi berjumlah lebih dari satu, maka bentuk umumnya menjadi:

<code>if (kondisi) {</code>
<code>statemen1;</code>
<code>statemen2;</code>
<code>...</code>
<code>}</code>

Perlu diperhatikan bahwa dalam PHP, kondisi harus berada di dalam tanda kurung. Jika tidak, maka program tidak bisa dijalankan.

Bentuk `if` seperti di atas hanya memiliki satu kondisi. Ini berarti bahwa jika kondisi tersebut tidak terpenuhi (bernilai `false`), maka statemen di dalam blok pemilihan tidak akan dilaksanakan dan eksekusi program akan

dilanjutkan ke statemen-statemen yang terdapat di bawah blok pemilihan tersebut.

Praktikum 6-1: Menggunakan statemen `if` dengan satu kondisi

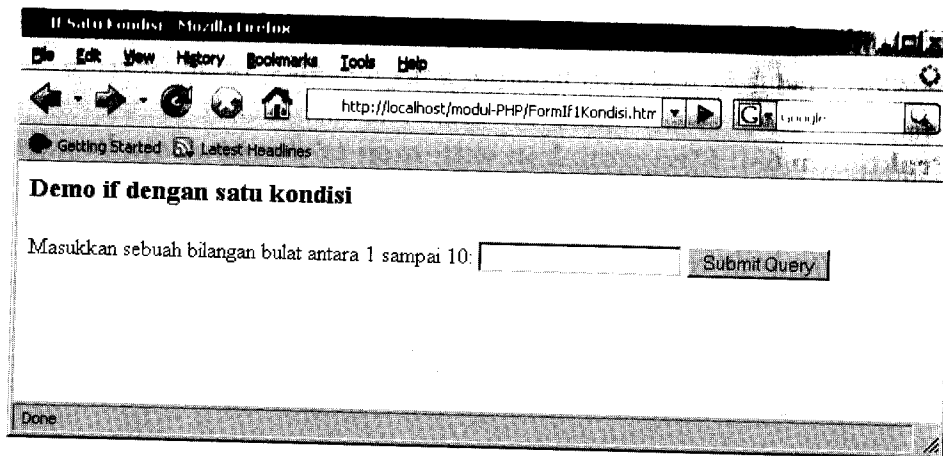
- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<html>
<head>
  <title>If Satu Kondisi</title>
</head>
<h3>Demo if dengan satu kondisi</h3>
<body>
  <form action="If1Kondisi.php" method="post">
    Masukkan sebuah bilangan bulat antara 1 sampai 10:
    <input type="text" name="angka">
    <input type="submit">
  </form>
</body>
</html>
```

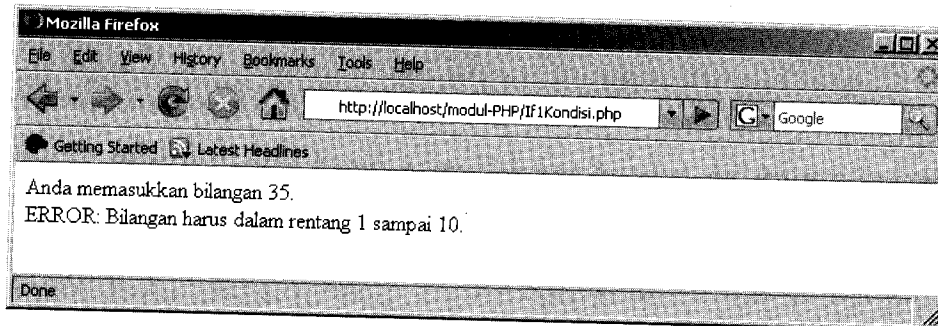
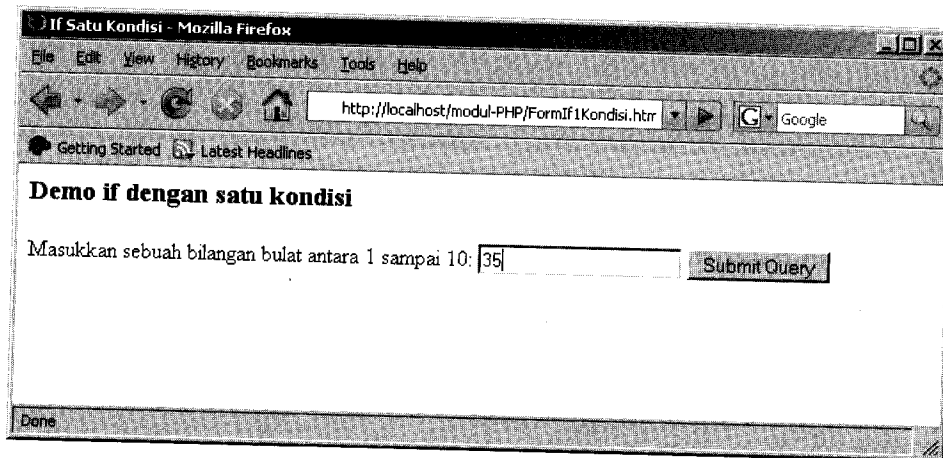
- ✓ Simpan *file* dengan nama **FormIf1Kondisi.html**
- ✓ Buatlah *file* baru, dan tuliskan kode berikut:

```
<?php
  $angka = $_POST["angka"];
  if ($angka < 1 || $angka > 10) {
    echo "Anda memasukkan bilangan $angka.<br />";
    echo "ERROR: Bilangan harus ".
      "dalam rentang 1 sampai 10.";
    exit();
  }
  echo "Bilangan yang Anda masukkan adalah $angka";
?>
```

- ✓ Simpan *file* dengan nama **If1Kondisi.php**
- ✓ Jalankan *web browser* dan masukkan alamat berikut:
<http://localhost/modul-PHP/FormIf1Kondisi.html>



- ☑ Isikan bilangan bulat yang berada di luar rentang (misalnya 35)



Keterangan:

Seperti yang Anda lihat pada hasil di atas bahwa pada saat kita mengisi bilangan yang lebih kecil dari 1 atau lebih besar dari 10, statemen yang terdapat di dalam blok pemilihan akan dilaksanakan. Sebaliknya, jika nilai yang dimasukkan berada dalam rentang 1 sampai 10 maka statemen-statemen tersebut akan diabaikan.

Jika kita ingin menyediakan statemen alternatif untuk kondisi yang tidak terpenuhi, maka Anda dapat menuliskannya setelah kata kunci **else**. Berikut ini bentuk umum penggunaannya.

```
if(kondisi){
    statemen1;
} else {
    statemen2;
}
```

statemen1 akan dieksekusi jika kondisi terpenuhi (bernilai true). Jika tidak terpenuhi, maka yang akan dieksekusi oleh program adalah statemen2.

Praktikum 6-2: Menggunakan statemen if dengan dua kondisi (Bagian 1)

- ☑ Jalankan aplikasi Text Editor
☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>If Dua Kondisi</title>
</head>
<h3>Demo ke-1 if dengan dua kondisi</h3>
<body>
    <form action="If2Kondisi1.php" method="post">
```



```

Masukkan sebuah bilangan bulat antara 1 sampai 10:
<input type="text" name="angka">
<input type="submit">
</form>
</body>
</html>

```

- ☑ Simpan file dengan nama **FormIf2Kondisi1.html**
- ☑ Buatlah file baru, dan tuliskan kode berikut:

```

<?php
$angka = $_POST["angka"];
if ($angka < 1 || $angka > 10) {
    echo "Anda memasukkan bilangan $angka. <br />";
    echo "ERROR: Bilangan harus ".
        "dalam rentang 1 sampai 10."
} else {
    echo "Bilangan yang Anda masukkan adalah $angka";
}
?>

```

- ☑ Simpan file dengan nama **If2Kondisi1.php**
- ☑ Jalankan web browser dan masukkan alamat berikut:
http://localhost/modul-PHP/FormIf2Kondisi1.html

Keterangan:

Hasil yang diperoleh akan sama seperti pada hasil yang diberikan oleh program pada **Praktikum 6-1**.

Praktikum 6-3: Menggunakan statemen if dengan dua kondisi (Bagian 2)

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```

<html>
<head>
    <title>If Dua Kondisi</title>
</head>
<h3>Demo ke-2 if dengan dua kondisi</h3>
<body>

```

```

<form action="If2Kondisi2.php" method="post">
    Masukkan sebuah bilangan bulat:
    <input type="text" name="bilangan">
    <input type="submit">
</form>
</body>
</html>

```

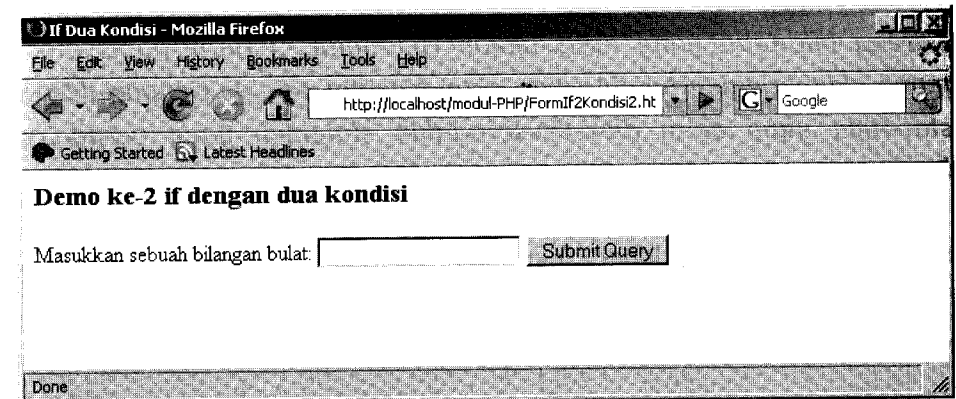
- ☑ Simpan file dengan nama **FormIf2Kondisi2.html**
- ☑ Buatlah file baru, dan tuliskan kode berikut:

```

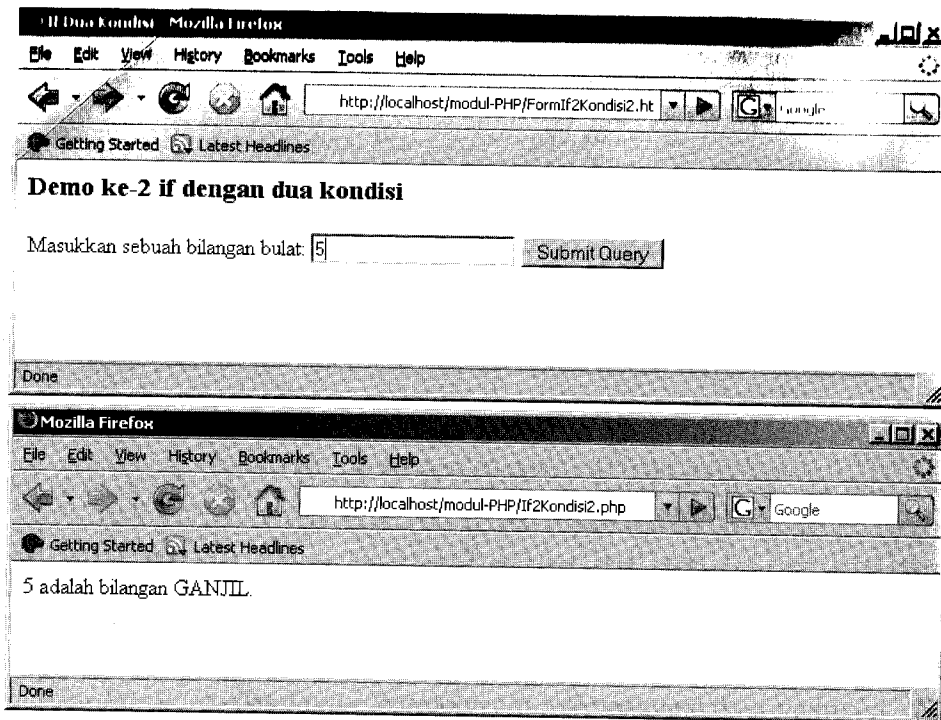
<?php
$bilangan = $_POST["bilangan"];
if ($bilangan % 2 == 0) {
    echo "$bilangan adalah bilangan GENAP. <br />";
} else {
    echo "$bilangan adalah bilangan GANJIL.";
}
?>

```

- ☑ Simpan file dengan nama **If2Kondisi2.php**
- ☑ Jalankan web browser dan masukkan alamat berikut:
http://localhost/modul-PHP/FormIf2Kondisi2.html



- ☑ Isikan dengan bilangan ganjil (misalnya 5)



Keterangan:

Karena bilangan yang dimasukkan (bilangan 5) tidak habis dibagi 2 maka kondisi yang didefinisikan akan bernilai *false*. Dengan demikian, statemen yang akan dieksekusi adalah statemen yang berada pada bagian *else*.

Statemen *if* juga dapat digunakan untuk memeriksa kondisi yang jumlahnya lebih dari dua. Caranya adalah dengan menggunakan klausa *elseif*. Berikut ini bentuk umum penggunaannya.

```
if (kondisi1) {
    statemen1;
} elseif (kondisi2) {
```

```
    statemen2;
} elseif (kondisi3) {
    statemen3;
} else {
    statemen4;
}
```

Statemen1 akan dieksekusi jika kondisi1 bernilai *true*. Jika tidak, blok pemilihan akan melanjutkan eksekusi dengan memeriksa kondisi2. Jika kondisi2 bernilai *true* maka program akan mengeksekusi statemen2. Jika tidak, program akan memeriksa kondisi3. Jika kondisi3 terpenuhi maka statemen3 akan dieksekusi. Jika semua kondisi tidak terpenuhi (bernilai *false*), maka yang akan dieksekusi oleh program adalah statemen yang terdapat pada bagian *else*, yaitu statemen4.

Dalam statemen *if* yang memiliki banyak kondisi seperti ini, klausa *else* harus diletakkan pada bagian paling akhir.

Praktikum 6-4: Menggunakan statemen *if* dengan banyak kondisi (Bagian 1)

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<html>
<head>
    <title>If Banyak Kondisi</title>
</head>
<h3>Demo ke-1 if dengan banyak kondisi</h3>
<body>
<form action="IfBanyakKondisi1.php" method="post">
    Masukkan sebuah bilangan:
    <input type="text" name="bilangan">
    <input type="submit">
```

```
</form>
</body>
</html>
```

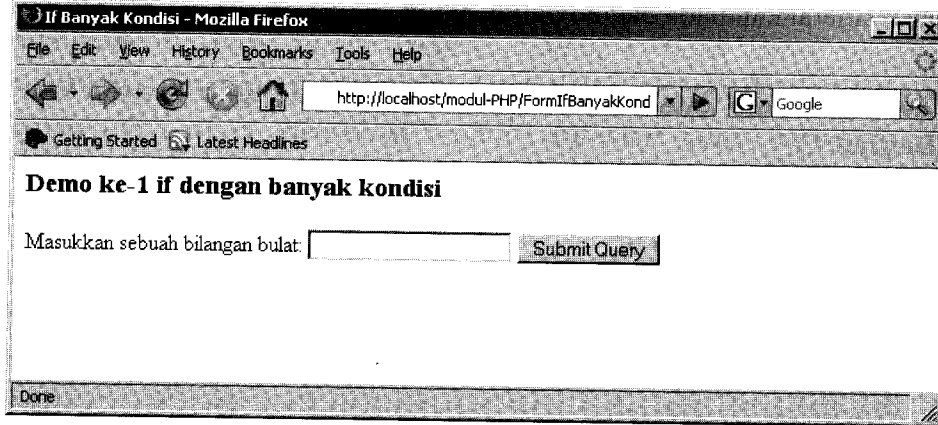
- ☑ Simpan *file* dengan nama FormIfBanyakKondisi1.html
- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```
<?php
    $bilangan = $_POST["bilangan"];

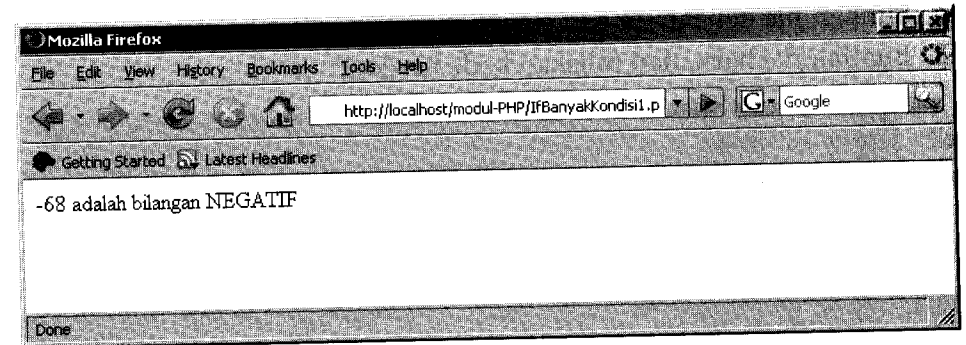
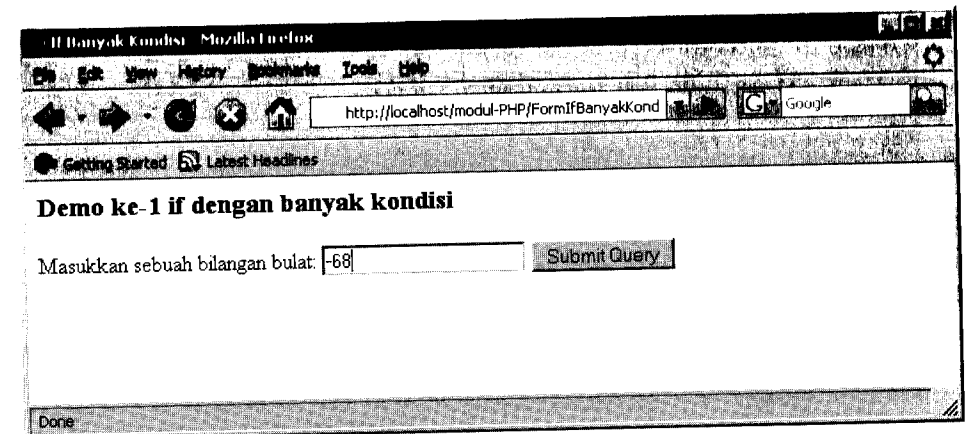
    if ($bilangan < 0) {
        $str = "NEGATIF";
    } elseif ($bilangan == 0) {
        $str = "NOL";
    } else {
        $str = "POSITIF";
    }

    echo "$bilangan adalah bilangan $str";
?>
```

- ☑ Simpan *file* dengan nama IfBanyakKondisi1.php
- ☑ Jalankan *web browser* dan masukkan alamat berikut:
http://localhost/modul-PHP/FormIfBanyakKondisi1.html



- ☑ Isikan bilangan negatif (misalnya -68)



Keterangan:

Karena nilai yang dimasukkan lebih kecil dari 0, maka kondisi pertama bernilai *true* sehingga statemen yang akan dieksekusi adalah statemen pada bagian pertama. Setelah mengeksekusi statemen tersebut, program akan menghentikan proses pemilihan.

Praktikum 6-4: Menggunakan statemen if dengan banyak kondisi (Bagian 2)

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
```

```

<title>If Banyak Kondisi</title>
</head>
<h3>Demo ke-2 if dengan banyak kondisi</h3>
<body>
<form action="IfBanyakKondisi2.php" method="post">
    Masukkan nilai Ujian Tengah Semester (UTS) :
    <input type="text" name="uts" /><br />
    Masukkan nilai Ujian Akhir Semester (UAS) :
    <input type="text" name="uas" /><br />
    <p><input type="submit" /></p>
</form>
</body>
</html>

```

- ☑ Simpan *file* dengan nama FormIfBanyakKondisi2.html
- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```

<?php
$uts = $_POST["uts"];
$uas = $_POST["uas"];

// menghitung nilai akhir, 40% UTS dan 60% UAS
$na = (0.4 * $uts) + (0.6 * $uas);

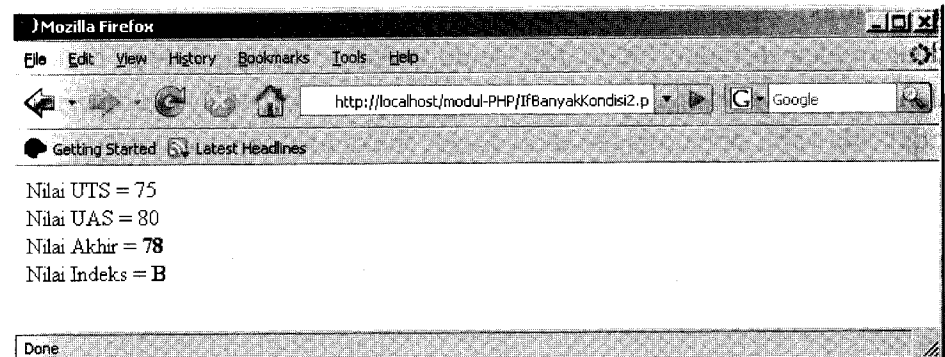
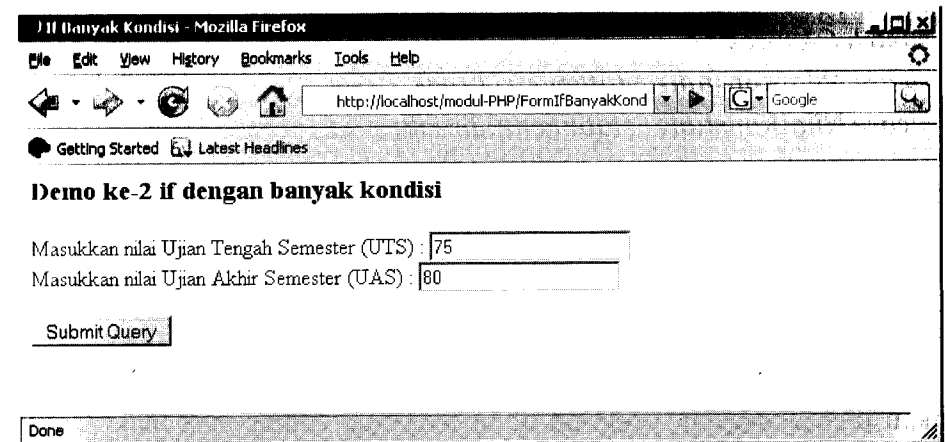
if ($na >= 80) {
    $indeks = 'A';
} elseif ($na >= 70) {
    $indeks = 'B';
} elseif ($na >= 50) {
    $indeks = 'C';
} elseif ($na >= 40) {
    $indeks = 'D';
} else {
    $indeks = 'E';
}

echo "Nilai UTS = $uts<br />";
echo "Nilai UAS = $uas<br />";
echo "Nilai Akhir = <strong>$na</strong><br />";
echo "Nilai Indeks = <strong>$indeks</strong>";
?>

```

- ☑ Simpan *file* dengan nama IfBanyakKondisi2.php

- ☑ Jalankan *web browser* dan masukkan alamat berikut:
<http://localhost/modul-PHP/FormIfBanyakKondisi2.html>
- ☑ Isikan nilai UTS dan UAS (misalnya 75 dan 80)



Keterangan:

Berdasarkan perhitungan dengan formula yang ditentukan di atas, variabel *\$na* akan bernilai 78. Dengan demikian, kondisi yang sesuai adalah kondisi kedua (indeks = B).

Menggunakan Statemen switch

Seperti halnya bahasa pemrograman C, C++, dan Java, PHP juga dilengkapi dengan adanya statemen switch. Statemen ini biasanya

digunakan untuk menggantikan struktur `if` yang memiliki terlalu banyak kondisi. Bentuk umum penggunaan statemen `switch` adalah sebagai berikut:

```
switch (ekspresi) {
    case NilaiKonstan1: statemen1; break;
    case NilaiKonstan2: statemen2; break;
    ...
    default:
        statemen alternatif;
}
```

Jika nilai ekspresi sama dengan `NilaiKonstan1`, maka `statemen1` akan dieksekusi dan program akan langsung keluar dari blok pemilihan. Jika tidak, program akan memeriksa `NilaiKonstan2`, apakah sama dengan ekspresi atau tidak. Jika sama maka `statemen2` akan dieksekusi, dan begitu seterusnya. Jika ternyata semua nilai konstan yang didefinisikan tidak ada yang sama dengan ekspresi, maka program akan mengeksekusi statemen alternatif (statemen yang terdapat pada bagian `default`).

Praktikum 6-5: Menggunakan statemen `if` untuk menentukan jumlah hari

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<?php
    $namafile = "IfJumlahHari.php";
?>

<html>
<head>
    <title>Statemen If</title>
</head>
<body>
<form action="<?php echo $namafile; ?>" method="post">
```

Pilih bulan:

```
<select name="bulan" size="1">
    <option value="0">[ none] </option>
    <option value="1">Januari</option>
    <option value="2">Februari</option>
    <option value="3">Maret</option>
    <option value="4">April</option>
    <option value="5">Mei</option>
    <option value="6">Juni</option>
    <option value="7">Juli</option>
    <option value="8">Agustus</option>
    <option value="9">September</option>
    <option value="10">Oktober</option>
    <option value="11">November</option>
    <option value="12">Desember</option>
</select>
<input type="submit" name="btnSubmit" />
</form>
```

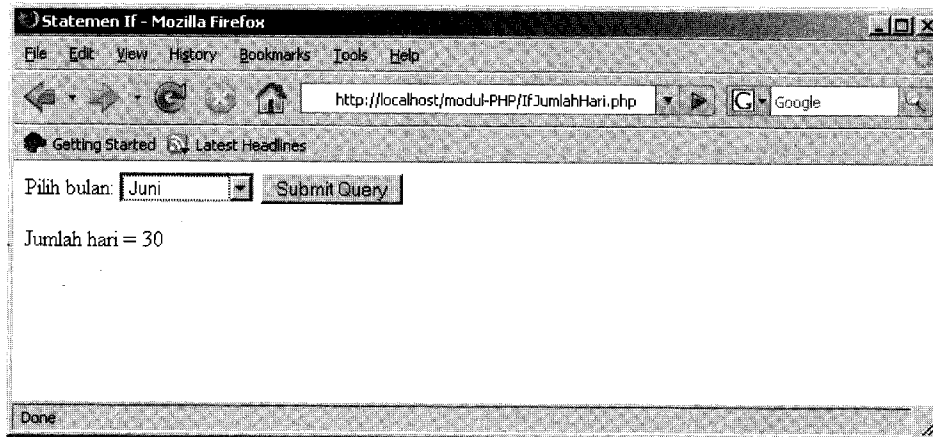
```
<?php
// menangani form dari file yang sama
if (isset($_POST["btnSubmit"])) {
    $bulan = $_POST["bulan"];
    if ($bulan == 1) {
        $jumlah = 31;
    } elseif ($bulan == 2) {
        $jumlah = 28; // untuk tahun kabisat,
        // $jumlah = 29
    } elseif ($bulan == 3) {
        $jumlah = 31;
    } elseif ($bulan == 4) {
        $jumlah = 30;
    } elseif ($bulan == 5) {
        $jumlah = 31;
    } elseif ($bulan == 6) {
        $jumlah = 30;
    } elseif ($bulan == 7) {
        $jumlah = 31;
    } elseif ($bulan == 8) {
        $jumlah = 31;
    } elseif ($bulan == 9) {
        $jumlah = 30;
    } elseif ($bulan == 10) {
        $jumlah = 31;
```

```

    } elseif ($bulan == 11) {
        $jumlah = 30;
    } elseif ($bulan == 12) {
        $jumlah = 31;
    } else {
        echo "Anda belum memilih nama bulan.";
        exit();
    }
    echo "Jumlah hari = $jumlah";
}
?>
</body>
</html>

```

- ☑ Simpan file dengan nama **IfJumlahHari.php**
- ☑ Jalankan web browser dan masukkan alamat berikut:
http://localhost/modul-PHP/IfJumlahHari.php



Keterangan:

Pada contoh program di atas, kita melakukan pemilihan dengan statemen *if*. Dalam kasus pemilihan yang melibatkan banyak kondisi seperti ini, sebenarnya kita dapat menyederhanakan kode di atas dengan penggunaan statemen *switch*.

Praktikum 6-6: Menggunakan statemen switch untuk menentukan jumlah hari

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```

<?php
    $namafile = "SwitchJumlahHari.php";
?>
<html>
<head>
    <title>Statemen switch</title>
</head>
<body>
<h3>Demo menggunakan statemen switch</h3>
<form action="<?php echo $namafile; ?>" method="post">
    Pilih bulan:
    <select name="bulan" size="1">
        <option value="0">[ none]</option>
        <option value="1">Januari</option>
        <option value="2">Februari</option>
        <option value="3">Maret</option>
        <option value="4">April</option>
        <option value="5">Mei</option>
        <option value="6">Juni</option>
        <option value="7">Juli</option>
        <option value="8">Agustus</option>
        <option value="9">September</option>
        <option value="10">Oktober</option>
        <option value="11">November</option>
        <option value="12">Desember</option>
    </select>
    <input type="submit" name="btnSubmit" />
</form>

<?php
    // menangani form dari file yang sama
    if (isset($_POST["btnSubmit"])) {
        $bulan = $_POST["bulan"];
        switch ($bulan) {
            case 1: $jumlah = 31; break;
            case 2: $jumlah = 28; break;

```

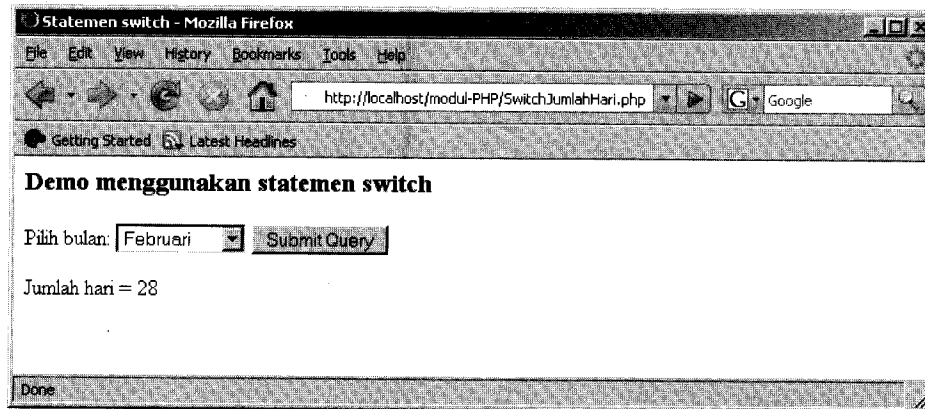
```

case 3: $jumlah = 31; break;
case 4: $jumlah = 30; break;
case 5: $jumlah = 31; break;
case 6: $jumlah = 30; break;
case 7: $jumlah = 31; break;
case 8: $jumlah = 31; break;
case 9: $jumlah = 30; break;
case 10: $jumlah = 31; break;
case 11: $jumlah = 30; break;
case 12: $jumlah = 31; break;
default:
    echo "Anda belum memilih bulan.";
    exit();
}
echo "Jumlah hari = $jumlah";
}
?>

</body>
</html>

```

- ☑ Simpan file dengan nama **SwitchJumlahHari.php**
- ☑ Jalankan *web browser* dan masukkan alamat berikut:
http://localhost/modul-PHP/SwitchJumlahHari.php



Keterangan:

Dengan menggunakan statemen *switch*, kode tampak lebih sederhana dan mudah untuk dibaca. Dalam kasus-kasus yang melibatkan banyak kondisi seperti ini, penggunaan statemen *switch* lebih direkomendasikan.

Untuk yang nilainya sama, dapat juga ditulis seperti ini:

```

switch ($bulan) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        $jumlah = 31; break;
    case 2:
        $jumlah = 28; break;
    case 4:
    case 6:
    case 9:
    case 11:
        $jumlah = 30; break;
    default:
        echo "Anda belum memilih bulan.";
        exit();
}

```

Statemen *break* dalam blok pemilihan *switch* digunakan untuk menghentikan pengulangan apabila telah ditemukan nilai konstan yang sesuai dengan ekspresi. Jika statemen *break* tidak disertakan, maka hasil dari program tidak akan sesuai dengan yang kita harapkan.

Praktikum 6-7: Menggunakan statemen *switch* tanpa statemen *break* (Bagian 1)

- ☑ Jalankan aplikasi Text Editor

- ☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>Switch tanpa break</title>
</head>
<body>
<h3>Demo ke-1 menggunakan switch tanpa break</h3>

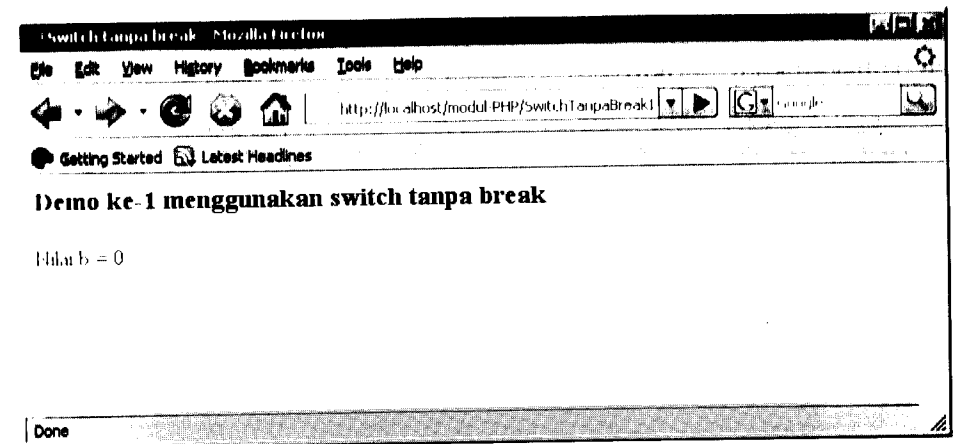
<?php
    $a = 2;

    switch($a) {
        case 1: $b = $a + 1;
        case 2: $b = $a + 2;
        case 3: $b = $a + 3;
        case 4: $b = $a + 4;
        case 5: $b = $a + 5;
        default:
            $b = 0;
    }

    echo "Nilai b = $b";
?>

</body>
</html>
```

- ☑ Simpan file dengan nama **SwitchTanpaBreak1.php**
- ☑ Jalankan web browser dan masukkan alamat berikut:
<http://localhost/modul-PHP/SwitchTanpaBreak1.php>



Keterangan:

Coba Anda perhatikan baik-baik: mengapa nilai *\$b* sama dengan 0? Bukan 4? Padahal nilai *\$a* sama dengan 2, yang berarti seharusnya program akan mengeksekusi statemen berikut:

```
$b = $a + 2;
```

Hal ini terjadi karena mula-mula program akan mengeksekusi statemen yang terdapat pada nilai konstan ke-2. Namun, karena setelah statemen tersebut tidak terdapat statemen *break*, maka program akan melanjutkan eksekusi terhadap nilai konstan 3, 4, 5, dan default. Pada bagian default, kita mengisi nilai 0 ke dalam variabel *\$b*. Maka dari itu, yang akan dipanggil oleh program adalah nilai *\$b* terakhir, yaitu 0.

Cara kerja dari kode program di atas sama seperti yang terdapat pada praktikum berikut ini.

Praktikum 6-8: Menggunakan statemen switch tanpa statemen break (Bagian 2)

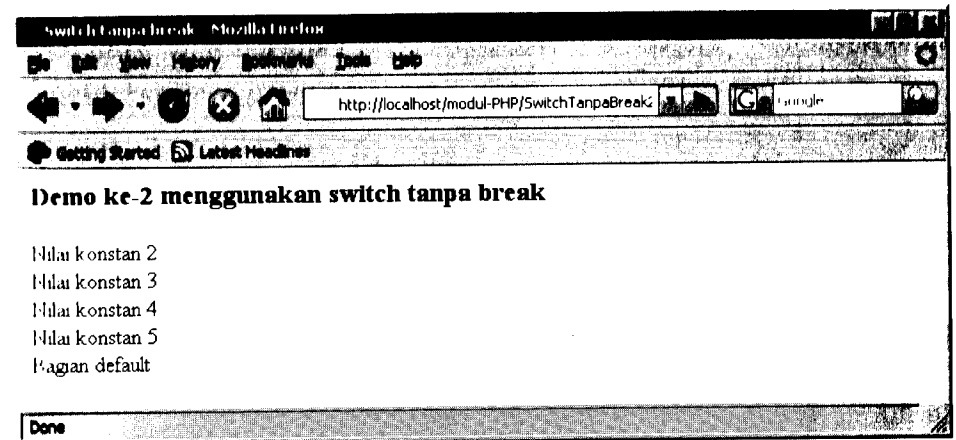
- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Switch tanpa break</title>
</head>
<body>
<h3>Demo ke-2 menggunakan switch tanpa break</h3>

<?php
  $a = 2;
  switch($a) {
    case 1: echo "Nilai konstan 1<br />";
    case 2: echo "Nilai konstan 2<br />";
    case 3: echo "Nilai konstan 3<br />";
    case 4: echo "Nilai konstan 4<br />";
    case 5: echo "Nilai konstan 5<br />";
    default:
      echo "Bagian default";
  }
?>

</body>
</html>
```

- ☒ Simpan file dengan nama **SwitchTanpaBreak2.php**
- ☒ Jalankan web browser dan masukkan alamat berikut:
http://localhost/modul-PHP/SwitchTanpaBreak2.php



Keterangan:

Dari hasil tersebut, Anda tentu dapat menyimpulkan betapa pentingnya penggunaan statemen *break* di dalam suatu pemilihan yang dibuat menggunakan statemen *switch*. Maka dari itu, jangan lupa untuk menyertakan statemen *break* pada setiap nilai konstan yang ada.

Ada hal menarik yang perlu disimak dalam penggunaan statemen *switch* di dalam PHP, yaitu nilai konstan yang bertipe *string*. Berbeda dengan bahasa pemrograman lain pada umumnya (seperti C/C++ dan Java), PHP membolehkan kita untuk memeriksa ekspresi yang bertipe *string*.

Praktikum 6-8: Menggunakan ekspresi *string* dalam statemen *switch*

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Statemen switch</title>
</head>
<body>
<h3>Demo menggunakan switch dengan ekspresi string</h3>
```

```

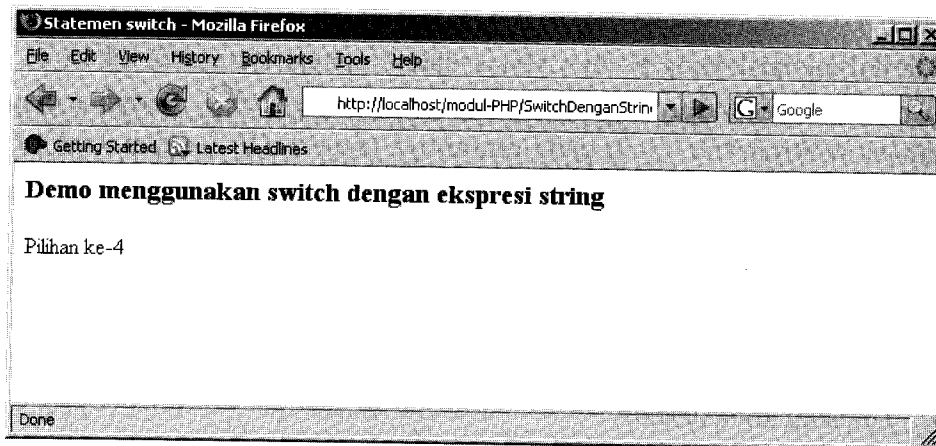
<?php
    $pilihan = "empat";

    switch ($pilihan) {
        case "satu": echo "Pilihan ke-1"; break;
        case "dua":  echo "Pilihan ke-2"; break;
        case "tiga": echo "Pilihan ke-3"; break;
        case "empat": echo "Pilihan ke-4"; break;
        case "lima": echo "Pilihan ke-5"; break;
        default:
            echo "Pilihan default";
    }
?>

</body>
</html>

```

- ☑ Simpan file dengan nama **SwitchDenganString.php**
- ☑ Jalankan web browser dan masukkan alamat berikut:
http://localhost/modul-PHP/SwitchDenganString.php



Keterangan:

Seperti yang tampak pada hasil program di atas, dalam PHP, Anda diizinkan untuk mendefinisikan nilai konstan yang bertipe string. Pada

kasus di atas, nilai variabel `$pilihan` bernilai "empat" sehingga yang akan dieksekusi oleh program adalah statemen:

```
echo "Pilihan ke-4"; break;
```

Mengulang Statemen

Dalam mengembangkan aplikasi *web*, kerap dijumpai kasus-kasus yang menuntut kita untuk melakukan pengulangan terhadap statemen-statemen tertentu. PHP menyediakan beberapa bentuk blok pengulangan untuk memudahkan kita, yaitu `while`, `do-while`, `for`, `foreach`.

Statemen *while*

Statemen `while` melakukan pemeriksaan kondisi di awal blok. Dengan demikian, jika ekspresi bernilai `false`, maka badan pengulangan akan diabaikan.

Bentuk umum:

```
while (ekspresi)
    Statemen;
```

// atau dapat juga ditulis:

```
while (ekspresi) {
    Statemen;
    ...
}
```

Contoh penggunaannya adalah sebagai berikut:

```

$i = 0;
while ($i < 10) {
    echo "Baris ke-" . $i . "<br />";
    $i++;
}

```

Kode di atas akan mengulang statemen sebanyak 10 kali, mulai indeks ke-0 sampai ke-9. Pengisian variabel `$i` dengan nilai 0 merupakan proses inisialisasi, sedangkan statemen `$i++` merupakan statemen *increment* agar nilai `$i` dinaikkan setiap kali statemen dalam blok pemilihan dieksekusi.

Cara kerja pengulangan `while` di atas adalah sebagai berikut:

1. Mula-mula (sebelum masuk ke badan pengulangan) variabel `$i` diinisialisasi dengan nilai 0.
2. Program akan memeriksa kondisi (`$i < 10`). Karena 0 lebih kecil dari 10 maka kondisi tersebut bernilai `true`, yang menyebabkan program akan mengeksekusi statemen di dalam badan pengulangan.
3. Mula-mula program akan mengeksekusi statemen berikut:

```
echo "Baris ke-" . $i . "<br />";
```

Di layar *web browser* akan tampil keluaran seperti berikut:

Baris ke-0

4. Program akan melanjutkan eksekusi ke baris selanjutnya:

```
$i++;
```

Eksekusi ini menyebabkan nilai variabel `$i` bertambah 1 sehingga nilai `$i` sekarang adalah 1.

5. Program akan kembali ke atas dengan memeriksa ulang kondisi (`$i < 10`). Karena 1 juga masih lebih kecil dari 10, maka kondisi pun bernilai

`true`. Dengan demikian, program akan kembali mengeksekusi statemen-statemen yang ada di dalam badan pengulangan.

Pada eksekusi baris pertama, hasil yang akan muncul di layar *web browser* adalah:

Baris ke-0

Baris ke-1

Pada eksekusi baris kedua, nilai `$i` akan dinaikkan lagi sebesar 1 sehingga nilai `$i` sekarang adalah 2.

6. Program akan kembali ke atas dengan memeriksa ulang kondisi (`$i < 10`). Selama nilai variabel `$i` masih lebih kecil dari 10, program akan terus mengeksekusi statemen-statemen yang ada di dalam badan pengulangan. Dengan demikian, hasil akhir yang akan diperoleh di layar *web browser* adalah:

Baris ke-0

Baris ke-1

Baris ke-2

Baris ke-3

Baris ke-4

Baris ke-5

Baris ke-6

Baris ke-7

Baris ke-8

Baris ke-9

7. Pada saat nilai `$i` sama dengan 10, kondisi (`$i < 10`) akan bernilai `false`. Hal ini menyebabkan program akan menghentikan proses pengulangan.

Praktikum 6-7: Menggunakan struktur pengulangan `while`

- ☑ Jalankan aplikasi Text Editor

- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Struktur pengulangan while</title>
</head>
<body>
<h3>Demo pengulangan menggunakan struktur while</h3>

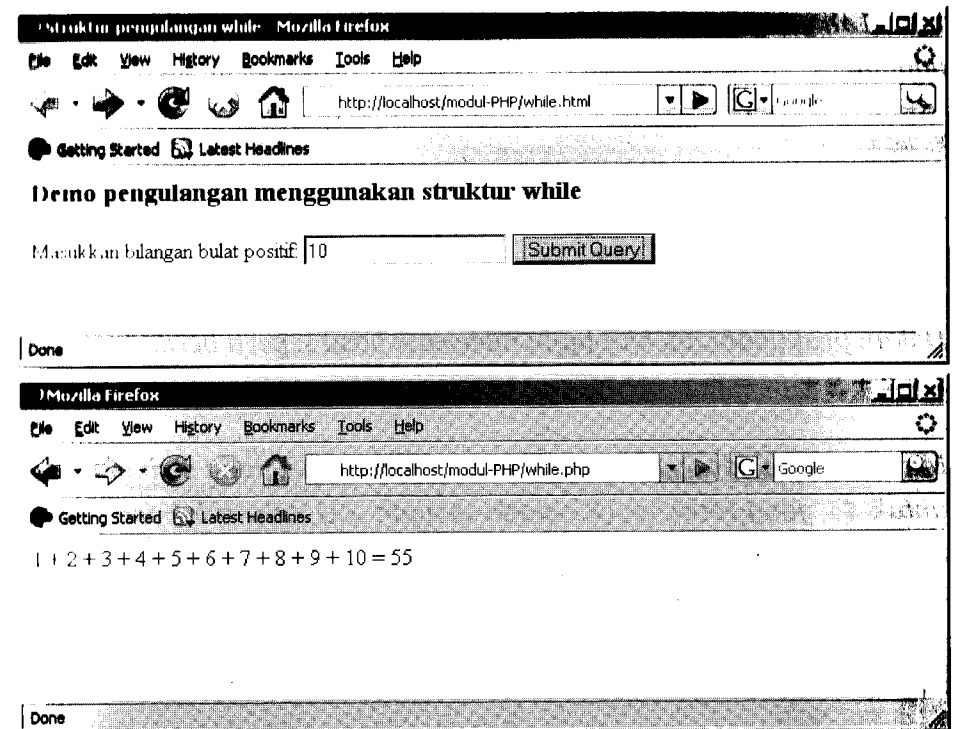
<form action="while.php" method="post">
  Masukkan bilangan bulat positif:
  <input type="text" name="n" value="10">
  <input type="submit">
</form>

</body>
</html>
```

- ☑ Simpan file dengan nama **while.html**
- ☑ Buatlah file baru, lalu tulis kode berikut:

```
<?php
$n = $_POST["n"];
if (isset($n)) {
  $jumlah = 0;
  $i = 1;
  while ($i <= $n) {
    if ($i != $n) {
      echo "$i + ";
    } else {
      echo "$i = ";
    }
    $jumlah += $i;
    $i++;
  }
  echo $jumlah;
}
?>
```

- ☑ Simpan file dengan nama **while.php**
- ☑ Jalankan web browser dan masukkan alamat berikut:
http://localhost/modul-PHP/while.html



Keterangan:

Program di atas berguna untuk menghitung jumlah dari n bilangan positif pertama. Adapun nilai n yang dimaksud dapat ditentukan oleh user melalui form yang disediakan.

Cara kerjanya sederhana, yaitu sebagai berikut: Mula-mula kita definisikan variabel $\$jumlah$, gunanya untuk menampung nilai total hasil perhitungan. Sebelum masuk ke badan pengulangan, nilai variabel $\$jumlah$ harus diset 0 karena memang belum diproses. Setelah itu, program akan mulai mengeksekusi statemen-statemen yang ada dalam proses pengulangan, dari $\$i$ sama dengan 1 sampai $\$i$ sama dengan n . Setiap kali proses eksekusi, nilai $\$jumlah$ akan ditambah dengan nilai $\$i$. Dengan demikian, nilai $\$jumlah$ sekarang sama dengan nilai $\$i$.

\$jumlah yang lama ditambah \$i. Perintah untuk melakukan hal tersebut adalah:

```
$jumlah += $i;
```

Kode di atas merupakan bentuk singkat dari:

```
$jumlah = $jumlah + $i;
```

Dalam praktikum ini, kita ingin mencetak hasil seperti berikut:

1 + 2 + 3 + ... = ?

Maka dari itu, kita perlu menyambung nilai \$i dengan string "+" kecuali untuk \$i yang terakhir (\$i sama dengan \$n), kita tidak menggunakan tanda "+", melainkan tanda "=". Adapun kode yang diperlukan untuk kebutuhan semacam ini adalah:

```
if ($i != $n) {  
    echo "$i + ";  
} else {  
    echo "$i = ";  
}
```

Setelah semua nilai \$i tercetak di layar web browser, langkah terakhir adalah mencetak nilai total hasil perhitungan, yaitu dengan cara:

```
echo $jumlah;
```

Statemen do-while

Berbeda dengan while, pada struktur pengulangan do-while kondisi akan diperiksa di bagian akhir blok. Hal ini mengakibatkan statemen dalam badan pengulangan minimal akan dieksekusi satu kali, meskipun kondisi yang didefinisikan ternyata tidak terpenuhi.

Bentuk umum:

```
do  
statemen;  
while (ekspresi);  
  
atau dapat ditulis:
```

```
do {  
    statemen;  
} while(ekspresi);
```

Contoh penggunaan:

```
$i = 0;  
while (1) {  
    echo "Baris ke-" . $i . "<br />";  
    $i++;  
} while($i < 10);
```

Melalui kode di atas, statemen dalam blok pengulangan akan dieksekusi terlebih dahulu sebelum kondisinya diperiksa.

Cara kerja struktur pengulangan do-while di atas adalah sebagai berikut:

1. Mula-mula nilai variabel \$i (variabel pencacah dalam pengulangan) diinisialisasi dengan nilai 0
2. Program akan masuk ke badan pengulangan dengan mengeksekusi statemen-statemen yang ada di dalamnya.

Pada saat eksekusi baris pertama, program akan mencetak nilai 0 ke layar:

Baris ke-0

Pada saat eksekusi baris kedua, nilai \$i akan dinaikkan sebesar 1, sehingga nilai \$i sekarang adalah 1.

3. Program akan memeriksa kondisi ($\$i < 10$). Karena nilai $\$i$ sama dengan 1 maka program akan kembali ke bagian awal pengulangan, dan kembali mengeksekusi statemen-statemen yang ada dalam badan pengulangan. Kali ini, karena $\$i$ bernilai 1, maka hasil yang akan diperoleh adalah:

Baris ke-0

Baris ke-1

Selanjutnya, nilai $\$i$ akan dinaikkan lagi menjadi 2. Begitu seterusnya sampai $\$i$ bernilai 9 sehingga hasil akhir yang akan diperoleh adalah:

Baris ke-0

Baris ke-1

Baris ke-2

Baris ke-3

Baris ke-4

Baris ke-5

Baris ke-6

Baris ke-7

Baris ke-8

Baris ke-9

4. Ketika $\$i$ bernilai 10, program akan menghentikan proses pengulangan karena kondisi ($\$i < 10$) sudah tidak terpenuhi lagi (bernilai false).

Praktikum 6-8: Menggunakan struktur pengulangan do-while

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Struktur pengulangan do-while</title>
</head>
<body>
```

h3>Demo pengulangan menggunakan struktur do-while</h3>

```
* form action="do-while.php" method="post">
  Masukkan bilangan bulat positif:
  * input type="text" name="n" value="10">
  * input type="submit">
* /form*

* /body*
* /html*
```

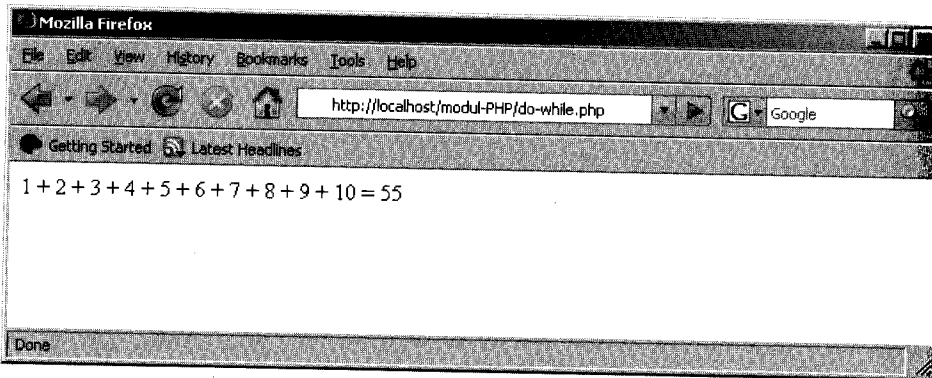
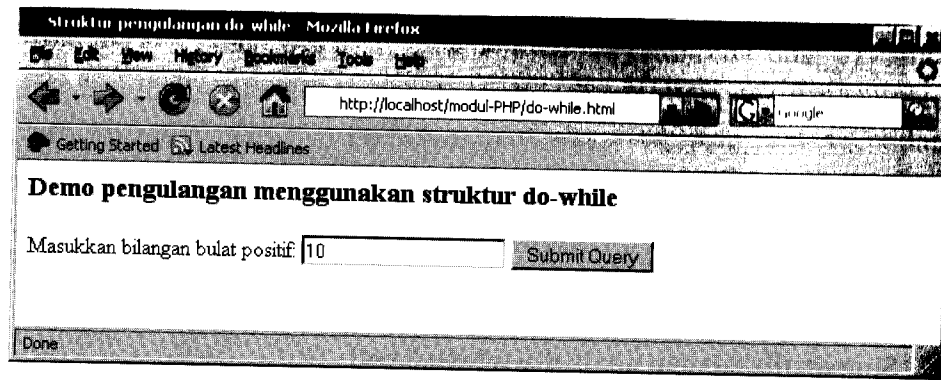
1. Simpan file dengan nama **do-while.html**

2. Buatlah file baru, lalu tulis kode berikut:

```
*?php
$n = $_POST["n"];
if (isset($n)) {
    $jumlah = 0;
    $i = 1;
    do {
        if ($i != $n) {
            echo "$i + ";
        } else {
            echo "$i = ";
        }
        $jumlah += $i;
        $i++;
    } while ($i <= $n);
    echo $jumlah;
}
?>
```

3. Simpan file dengan nama **do-while.php**

4. Jalankan web browser dan masukkan alamat berikut:
<http://localhost/modul-PHP/do-while.html>



Keterangan:

Seperti yang Anda lihat, hasil yang diperoleh ternyata dengan pengulangan yang dilakukan dengan struktur *while*. Perbedaan yang mendasar adalah masalah penempatan kondisinya saja. Pada struktur *while*, pemeriksaan kondisi dilakukan di awal; sedangkan pada struktur *do-while*, pemeriksaan kondisi ditempatkan pada bagian akhir.

Statemen *for*

Statemen *for* dapat dikatakan sebagai bentuk pengulangan paling sederhana di antara bentuk lainnya.

Bentuk umum:

```
for (inisialisasi; ekspresi; increment/decrement)
    statemen;
```

// atau dapat juga ditulis:

```
for (inisialisasi; ekspresi; increment/decrement) {
    statemen;
}
```

Contoh penggunaan:

```
for ($i=0; $i<10; $i++) {
    echo "Baris ke-" . $i . "<br />";
}
```

Baris kode di atas akan mencetak sepuluh buah baris teks di layar *web browser*, dimulai dari indeks ke-0 sampai ke-9.

Cara kerja struktur pengulangan *for* di atas adalah sebagai berikut:

1. Mula-mula program akan melakukan inisialisasi terhadap nilai *\$i* dengan mengeksekusi bagian inisialisasi, yaitu:

```
$i = 0;
```

Perlu Anda ketahui bahwa bagian ini hanya akan dieksekusi sekali saja.

2. Program akan memeriksa kondisi *\$i < 10*. Karena nilai *\$i* masih 0, yang berarti lebih kecil dari 10, maka kondisi bernilai *true*. Dan karena kondisi bernilai *true*, maka program akan mengeksekusi statemen yang ada dalam badan pengulangan.

Di layar *web browser* akan tampil:

Baris ke-0

3. Setelah selesai mengeksekusi statemen yang ada dalam badan pengulangan, program akan mengeksekusi statemen *increment*, yaitu:

```
$i++
```

Dengan demikian, nilai `$i` sekarang adalah 1.

4. Program akan kembali memeriksa kondisi `$i < 10`. Karena 1 lebih kecil dari 10, program akan kembali masuk ke badan pengulangan dengan mengeksekusi statemen di dalamnya.

Pada saat ini, di layar *web browser* akan tampil:

Baris ke-0

Baris ke-1

5. Proses ini akan terus diulang sampai ditemukan kondisi *false* (dalam kasus ini pada saat `$i` bernilai 10). Dengan demikian, hasil akhir yang diperoleh di layar *web browser* adalah:

Baris ke-0

Baris ke-1

Baris ke-2

Baris ke-3

Baris ke-4

Baris ke-5

Baris ke-6

Baris ke-7

Baris ke-8

Baris ke-9

6. Pada saat `$i` bernilai 10, program akan menghentikan proses pengulangan.

Praktikum 6-8: Menggunakan struktur pengulangan **for**

- ☒ Jalankan aplikasi Text Editor

- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Struktur pengulangan for</title>
</head>
<body>
  <h3>Demo pengulangan menggunakan struktur for</h3>

  <form action="for.php" method="post">
    Masukkan bilangan bulat positif:
    <input type="text" name="n" value="10">
    <input type="submit">
  </form>

</body>
</html>
```

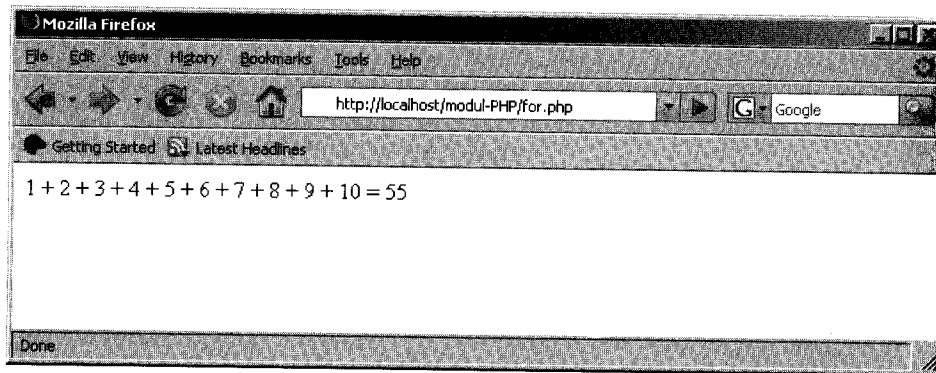
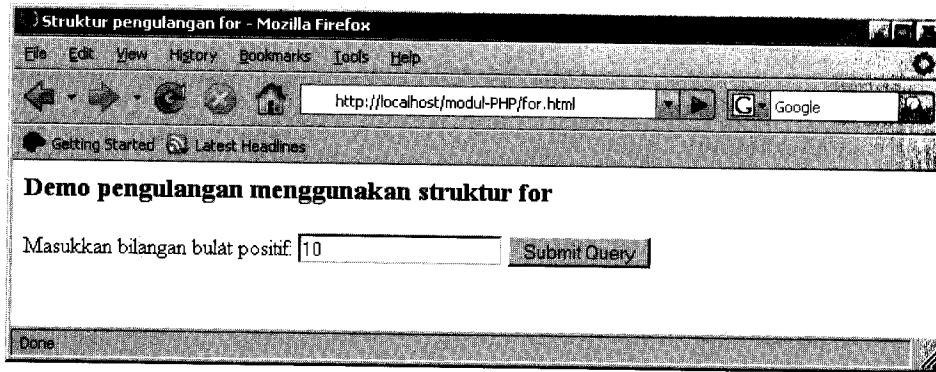
- ☒ Simpan *file* dengan nama **for.html**

- ☒ Buatlah *file* baru, lalu tulis kode berikut:

```
<?php
$n = $_POST["n"];
if (isset($n)) {
  $jumlah = 0;
  for ($i=1; $i <= $n; $i++) {
    if ($i != $n) {
      echo "$i + ";
    } else {
      echo "$i = ";
    }
    $jumlah += $i;
  }
  echo $jumlah;
}
?>
```

- ☒ Simpan *file* dengan nama **for.php**

- ☑ Jalankan *web browser* dan masukkan alamat berikut:
http://localhost/modul-PHP/for.html



Keterangan:

Dari kode di atas dapat Anda lihat bahwa kode pengulangan menggunakan struktur *for* sangat sederhana, dimana bagian inisialisasi, pemeriksaan kondisi, dan increment/decrement terkumpul menjadi satu baris, meskipun eksekusinya tidak dilakukan secara berurutan. Sebenarnya untuk melakukan pengulangan *statemen* Anda dapat menggunakan bentuk yang mana saja, tergantung pada selera dan pilihan Anda, yang penting algoritmanya benar.

Statemen *foreach*

Statemen *foreach* digunakan untuk melakukan pengulangan pada saat mengakses elemen-elemen dalam suatu *array*.

Bentuk umum:

```
foreach ($array as $current)
    statemen;
```

// atau dapat juga ditulis:

```
foreach ($array as $current) {
    statemen;
}
```

Contoh penggunaannya adalah sebagai berikut:

```
$pegawai = array('manajer' => 'Aas',
                 'dirut'    => 'Jajang',
                 'umum'     => 'Dadan',
                 'teknisi'  => 'Renra');
```

```
foreach ($pegawai as $nama) {
    echo "Hallo, $nama<br />";
}
```

atau:

```
foreach ($pegawai as $jabatan => $pejabat) {
    echo "$jabatan dijabat oleh $pejabat";
}
```

Pada kode di atas, kita mendeklarasikan *array* *\$pegawai* yang berisi empat elemen. *Array* di atas memiliki indeks bertipe *string*, sehingga dapat diakses melalui cara berikut:

```
$nama = $pegawai['manajer'];
echo $nama . "<br />";
$nama = $pegawai['dirut'];
echo $nama . "<br />";
$nama = $pegawai['umum'];
echo $nama . "<br />";
$nama = $pegawai['teknisi'];
```

Untuk memudahkan dalam menyelesaikan proses di atas, PHP menyediakan statemen `foreach`. Statemen `foreach` akan membaca seluruh elemen yang ada satu demi satu. Dengan demikian, cara aksesnya dapat disederhanakan menjadi seperti berikut:

```
foreach($pegawai as $nama) {
    echo "Hallo, $nama<br />";
}
```

Praktikum 6-8: Menggunakan struktur pengulangan `foreach`

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
    <title>Struktur pengulangan foreach</title>
</head>
<body>
<h3>Demo pengulangan menggunakan struktur foreach</h3>

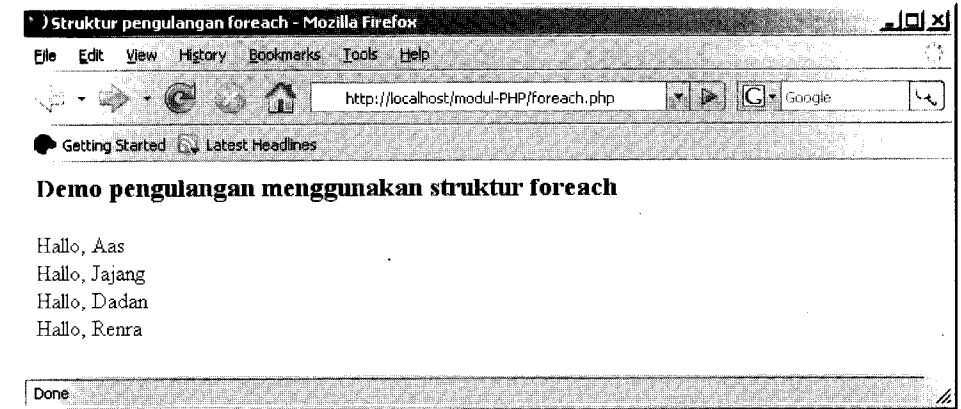
<?php
$pegawai = array('manajer' => 'Aas',
                 'dirut'    => 'Jajang',
                 'umum'     => 'Dadan',
                 'teknisi' => 'Renra');

foreach($pegawai as $nama) {
    echo "Hallo, $nama<br />";
}

?>
```

```
• /body •
• /html •
```

- 1. Simpan file dengan nama **foreach.php**
- 2. Jalankan *web browser* dan masukkan alamat berikut:
http://localhost/modul-PHP/foreach.php



Keterangan:

Mula-mula statemen `foreach` akan membaca elemen pertama array `$pegawai` dan menyimpan nilainya ke dalam variabel `$nama`. Selanjutnya, program akan mencetak statemen:

```
echo "Hallo, $nama<br />";
```

Melalui kode tersebut, di layar *web browser* akan muncul hasil seperti berikut:

Hallo, Aas

Selanjutnya program akan melanjutkan proses pembacaan elemen array ke elemen berikutnya, yaitu elemen kedua. Setelah dibaca, isi atau nilai dari elemen array kedua pun ditampung kembali ke variabel `$nama`. Dengan demikian, hasil di layar *web browser* menjadi seperti berikut:

Hallo, Aas
Hallo, Jajang

Proses seperti di atas akan diulang secara terus menerus sampai jumlah elemen array terbaca semua.



Membuat Fungsi

Fungsi adalah blok program yang didefinisikan dengan nama tertentu dan digunakan untuk melaksanakan tugas-tugas spesifik. Berapa kali pun kita memanggil suatu fungsi, fungsi tersebut hanya akan dikompilasi sekali oleh *compiler*. Fungsi berguna untuk membuat kode program lebih bersifat modular dan lebih mudah dibaca. Dalam modul ini kita akan membahas dan mempraktikkan materi-materi berikut:

- Mengetahui jenis fungsi dalam PHP
- Cara mendefinisikan sebuah fungsi
- Cara memanggil fungsi

- Melewatkan parameter ke dalam fungsi
- Membuat parameter *default*

Jenis Fungsi dalam PHP

Dalam PHP, fungsi dibedakan menjadi 2 (dua) jenis, yaitu:

- Fungsi yang tidak mengembalikan nilai (sering disebut *void function*)
- Fungsi yang mengembalikan nilai

Ciri fungsi yang mengembalikan nilai adalah, dalam fungsi tersebut selalu terdapat kata kunci `return` (biasanya tertulis pada bagian/baris terakhir).

Contoh fungsi yang tidak mengembalikan nilai:

```
function Header() {
    echo "<html>";
    echo "<head>";
    echo "<title>Test Function</title>";
    echo "</head>";
}
```

Contoh fungsi yang mengembalikan nilai:

```
function gabungString($kiri, $kanan) {
    $gabungan = $kiri . $kanan;
    return $gabungan;
}
```

Jika sebelumnya Anda pernah mempelajari bahasa pemrograman lain:

- Dalam bahasa Pascal, fungsi yang tidak mengembalikan nilai disebut **prosedur**.
- Dalam bahasa C/C++, fungsi yang tidak mengembalikan nilai disebut ***void function***.

- Dalam bahasa Java, fungsi yang tidak mengembalikan nilai disebut ***void method***.

Cara pemanggilan kedua jenis fungsi di atas pun berbeda. Kita akan membahas tentang hal tersebut pada bagian selanjutnya.

Mendefinisikan dan Memanggil Fungsi

Untuk mendefinisikan suatu fungsi di dalam PHP, gunakan bentuk umum berikut:

```
function [&] NamaFungsi ([parameter, ...]) {
    // Daftar statemen
    ...
}
```

Daftar statemen di atas tidak harus berupa kode PHP, bisa juga berbentuk kode HTML. Sebagai contoh:

```
// function column() { ?>
    </td><td>
? } ?>
```

Badan fungsi di atas hanya berisi kode HTML murni, bukan PHP. Adapun contoh fungsi yang menggunakan kode PHP murni adalah sebagai berikut:

```
function gabungString($kiri, $kanan) {
    $gabungan = $kiri . $kanan;
    return $gabungan;
}
```

Fungsi di atas memiliki dua buah parameter: `$kiri` dan `$kanan`. Dengan menggunakan operator penyambung *string* (dalam PHP operator tersebut

direpresentasikan dengan **tanda titik**), kedua nilai yang dilewatkan melalui parameter tersebut akan digabungkan. Variabel \$gabungan merupakan variabel lokal yang berfungsi sebagai variabel temporer, untuk memudahkan proses pembacaan algoritma.

Sebenarnya fungsi di atas dapat disederhanakan lagi menjadi seperti berikut:

```
function gabungString($kiri, $kanan) {
    return $kiri . $kanan;
}
```

Pada pendefinisian fungsi kali ini, tidak terdapat adanya variabel lokal. Artinya, di dalam badan fungsi, nilai parameter akan diproses dan hasilnya langsung dikembalikan.

Untuk menggunakan fungsi yang telah didefinisikan di atas, kita perlu memanggil fungsi tersebut. Adapun cara pemanggilannya adalah sebagai berikut:

- Untuk fungsi yang tidak mengembalikan nilai, cukup tuliskan nama fungsinya dan daftar parameternya (bila ada).

Contoh:

```
Header();           // memanggil fungsi Header()
column();           // memanggil fungsi column()
```

- Untuk fungsi yang mengembalikan nilai, sediakan variabel bantu untuk menampung nilai kembalian fungsi. Bisa juga dengan langsung memproses hasil dari fungsi tersebut.

Contoh:

```
// menggunakan variabel bantu untuk penampung
$hasil = gabungString("Ini adalah ", "kalimat lengkap.");
echo $hasil;
```

```
// Nilai kembalian fungsi langsung diproses
echo gabungString("Ini adalah ", "kalimat lengkap.");
```

Praktikum 7-1: Mendefinisikan dan Memanggil Fungsi (Bagian 1)

- 1. Jalankan aplikasi Text Editor
- 2. Tuliskan kode berikut:

```
<html>
<head>
    <title>Mendefinisikan dan Memanggil Fungsi</title>
</head>
<body>

<?php

function judul() {
    echo "<h2>Demo ke-1 mendefinisikan " .
        "dan memanggil fungsi</h2>";
}

function gabungString($kiri, $kanan) {
    return $kiri . $kanan;
}

$str1 = "Ini adalah ";
$str2 = "kalimat lengkap.";

$hasil = gabungString($str1, $str2);

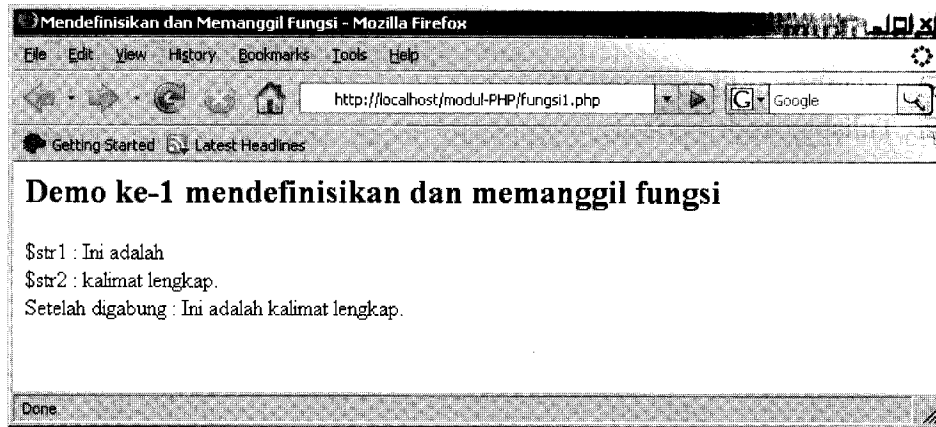
judul();

echo "\\ $str1 : $str1<br />";
echo "\\ $str2 : $str2<br />";
echo "Setelah digabung : $hasil";

?>
</body>
</html>
```

- 3. Simpan file dengan nama **fungsi1.php**

- ☑ Jalankan *web browser* dan masukkan <http://localhost/modul-PHP/fungsi1.php>



Keterangan:

Pada praktikum di atas kita mendefinisikan dua buah fungsi.

Pertama, `judul()`, adalah fungsi yang tidak mengembalikan nilai. Fungsi tersebut juga tidak memiliki parameter. Tugasnya hanya untuk mencetak teks `judul` ke layar *web browser*. Cara pemanggilan fungsi pertama sangatlah sederhana, yaitu:

```
judul();
```

Fungsi kedua, `gabungString()`, adalah fungsi yang mengembalikan nilai. Fungsi tersebut memiliki dua buah parameter. Oleh karena itu, pada saat pemanggilan, kita juga harus menyertakan dua buah argumen (nilai parameter) ke dalam fungsi tersebut.

```
$hasil = gabungString($str1, $str2);
```

Variabel `$hasil` digunakan untuk menampung nilai yang dikembalikan oleh fungsi `gabungString()`.

Praktikum 7-2: Mendefinisikan dan Memanggil Fungsi (Bagian 2)

- ☑ Jalankan aplikasi *Text Editor*
- ☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>Mendefinisikan dan Memanggil Fungsi</title>
</head>
<body>

<?php

function judul() {
    echo "<h2>Demo ke-2 mendefinisikan " .
        "dan memanggil fungsi</h2>";
}

function KaliDua($nilai) {
    return $nilai << 1;
}

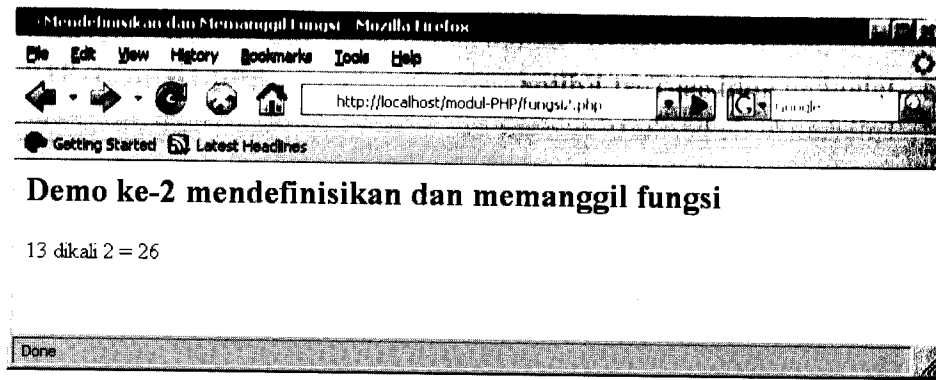
judul();

$hasil = KaliDua(13);

echo "13 dikali 2 = $hasil";

?>
</body>
</html>
```

- ☑ Simpan file dengan nama **fungsi2.php**
- ☑ Jalankan *web browser* dan masukkan <http://localhost/modul-PHP/fungsi2.php>



Keterangan:

Dalam praktikum ini kita juga mendefinisikan dua buah fungsi. Fungsi kedua, *KaliDua()*, digunakan untuk mengalikan suatu bilangan dengan 2. Operator yang digunakan di dalam fungsi tersebut adalah operator **bitwise shift left**, yang berfungsi untuk menggeser bit (binary digit) ke arah kiri. Setiap pergeseran bit ke arah kiri akan menghasilkan dua kali angka semula.

```
$nilai << 1;
```

Angka 1 di atas menunjukkan penggeseran sebanyak 1 bit.

Pada saat pemanggilan, kita melewati nilai 13 sebagai argumennya sehingga nilai yang akan dihasilkan oleh fungsi *KaliDua()* adalah 26.

Fungsi Bersarang (Nested Function)

PHP juga mendukung fungsi bersarang (*nested function*) atau fungsi di dalam fungsi. Berikut ini contoh pendefinisianannya:

```
function outer ($a) {
    function inner ($b) {
        echo "there $b";
    }
}
```

```
echo "$a, hello ";
}
outer("well");
inner("reader");
```

Meskipun pada kenyataannya *nested function* ini jarang digunakan, tapi pada kasus-kasus tertentu mungkin saja Anda membutuhkannya.

Melewatkan Parameter ke dalam Fungsi

Parameter adalah nilai yang dilewatkan ke dalam suatu fungsi. Gunanya agar fungsi tersebut dapat memperoleh nilai dinamis sesuai argumen yang diisi pada saat pemanggilan fungsi tersebut.

Terdapat dua cara melewati parameter, yaitu berdasarkan nilai dan berdasarkan referensinya.

Pass by Value

Dalam banyak kasus, Anda akan melewati parameter berdasarkan nilai (*pass by value*). Argumen yang dilewatkan harus berupa ekspresi yang valid. Ekspresi tersebut selanjutnya akan diproses di dalam fungsi dan hasil prosesnya (biasanya berupa nilai) akan dikembalikan sebagai *return value*.

```
function KaliDua($nilai) {
    $nilai = $nilai << 1;
    return $nilai;
}
```

Fungsi di atas dapat menunjukkan bagaimana suatu parameter dilewatkan berdasarkan nilainya.

Praktikum 7-3: Melewatkan parameter berdasarkan nilai (pass by value)

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<html>
<head>
  <title>Melewatkan parameter</title>
</head>
<body>

<?php

function judul() {
  echo "<h2>Demo melewati parameter berdasarkan " .
    "nilai (pass by value)</h2>";
}

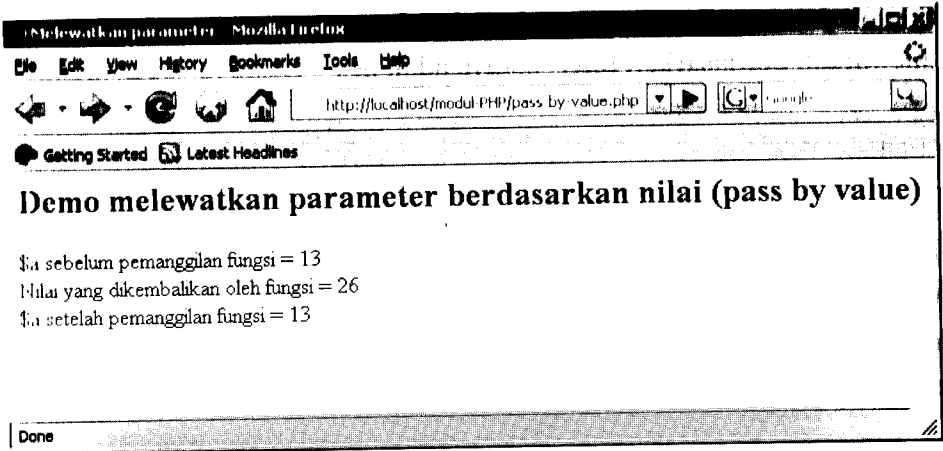
function KaliDua($nilai) {
  $nilai = $nilai << 1;
  return $nilai;
}

judul();

$a = 13;
echo "\$a sebelum pemanggilan fungsi = $a<br />";
echo "Nilai yang dikembalikan oleh fungsi = " .
  KaliDua($a) . "<br />";
echo "\$a setelah pemanggilan fungsi = $a";

?>
</body>
</html>
```

- ✓ Simpan file dengan nama **pass-by-value.php**
- ✓ Jalankan web browser dan masukkan:
http://localhost/modul-PHP/pass-by-value.php



Keterangan:

Seperti yang Anda lihat di atas, variabel yang digunakan sebagai parameter pada saat pemanggilan (\$a), ternyata nilainya tidak ikut berubah setelah pemanggilan fungsi. Hal ini disebabkan nilai dari variabel \$a akan disalin (copy) ke variabel \$nilai. Karena dua variabel ini (\$a dan \$nilai) tidak ada kaitannya, maka yang terjadi pada variabel \$nilai tentu tidak akan mempengaruhi nilai variabel \$a.

Pass by Reference

Dalam *pass by reference*, argumen harus berupa variabel. Selain itu, pada saat pendefinisian fungsinya, kita harus membubuhkan tanda *ampersand* (&) di depan parameter bersangkutan.

Berikut ini contoh kode yang menunjukkan cara melewati parameter berdasarkan referensi.

```
function KaliDua(&$nilai) {
  $nilai = $nilai << 1;
  return $nilai;
}
```


Contoh pemanggilan fungsi yang di dalamnya mengandung parameter yang dilewatkan berdasarkan referensinya adalah:

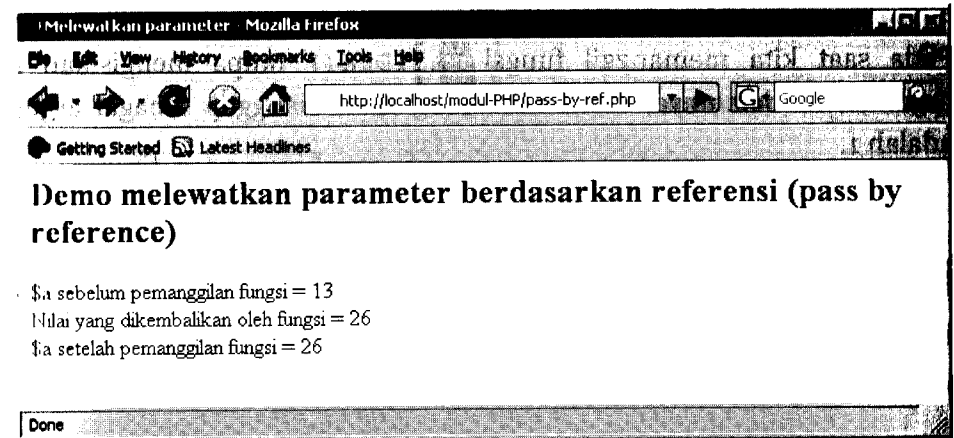
```
$a = 3;  
$KaliDua($a);  
echo $a
```

Praktikum 7-4: Melewatkan parameter berdasarkan referensi (*pass by reference*)

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>  
<head>  
  <title>Melewatkan parameter</title>  
</head>  
<body>  
  
<?php  
  
function judul() {  
  echo "<h2>Demo melewati parameter berdasarkan "  
    "referensi (pass by reference)</h2>";  
}  
  
function KaliDua(&$nilai) {  
  $nilai = $nilai << 1;  
  return $nilai;  
}  
  
judul();  
  
$a = 13;  
echo "\$a sebelum pemanggilan fungsi = $a<br />";  
echo "Nilai yang dikembalikan oleh fungsi = "  
  KaliDua($a) . "<br />";  
echo "\$a setelah pemanggilan fungsi = $a";  
  
?>  
</body>  
</html>
```

- M Simpan file dengan nama **pass-by-ref.php**
- M Jalankan web browser dan masukkan **http://localhost/modul-PHP/pass-by-ref.php**



Keterangan:

Pada praktikum kali ini, kita mendefinisikan fungsi yang menggunakan parameter berupa referensi (dengan penambahan tanda & di depan parameter bersangkutan). Ini berarti bahwa yang akan dikirimkan ke dalam fungsi adalah bukan berupa nilai, tapi lokasi atau alamat tertentu. Pada kasus di atas, *\$nilai* akan mengacu ke variabel *\$a*. Dengan kata lain, keduanya akan menunjuk ke alamat yang sama. Dengan demikian, perubahan yang terjadi pada variabel *\$nilai* juga akan mempengaruhi nilai variabel *\$a*. Hal ini terjadi karena sebenarnya kedua variabel tersebut adalah sama, tapi memiliki dua nama.

Membuat Parameter Default

Parameter *default* adalah parameter yang pada saat dilewatkan telah memiliki nilai *default*. Untuk menentukan nilai *default* tersebut, gunakan operator *assignment* pada saat Anda mendeklarasikan fungsi.

Contoh:

```
function KaliDua($nilai = 1) {  
    return $nilai << 1;  
}
```

Pada saat kita memanggil fungsi `KaliDua()` tanpa mengisi nilai argumen ke dalamnya, maka nilai yang akan dianggap sebagai argumen adalah 1.

Nilai yang Dikembalikan oleh Fungsi (*Return Value*)

Fungsi di dalam PHP dapat mengembalikan nilai tunggal maupun nilai yang jumlahnya lebih dari satu.

Fungsi yang mengembalikan nilai tunggal:

```
function return_one() {  
    return 30;  
}
```

Fungsi yang mengembalikan nilai lebih dari satu:

```
function return_multiple() {  
    return array("Pras", 46);  
}
```

Bekerja dengan *Array*

Array adalah suatu koleksi nilai yang saling berkaitan. Pada modul sebelumnya Anda telah mengenal arti variabel, yaitu suatu pengenal yang dapat menyimpan sebuah nilai. *Array*, fungsinya sama seperti variabel, hanya saja nilai yang ditampung di dalamnya bisa banyak, tidak hanya satu. *Array* tersusun atas elemen-elemen. Setiap elemen memiliki kunci (*key*) dan nilai (*value*). Dalam modul ini Anda akan mempelajari materi-materi berikut:

- Mengetahui jenis *array* dalam PHP
- Cara membuat *array*

- Cara mengakses elemen-elemen atau nilai nilai yang tersimpan di dalam *array*
- Menggunakan pengulangan untuk mengakses elemen-elemen *array*
- Memodifikasi *array*
- Mengurutkan *array*

Jenis Array dalam PHP

Dalam PHP, array diklasifikasikan menjadi 2 (dua) jenis, yaitu:

- *Array* berindeks (*indexed array*)
- *Array* asosiatif (*associative array*)

Array Berindeks

Array berindeks adalah *array* yang diindeks menggunakan angka/nomor maupun karakter yang sifatnya berurutan.

Coba Anda perhatikan gambar berikut:

101	88	100	90	70
<code>\$A[0]</code>	<code>\$A[1]</code>	<code>\$A[2]</code>	<code>\$A[3]</code>	<code>\$A[4]</code>

Gambar di atas merupakan visualisasi dari *array* \$A yang memiliki lima buah elemen. 101, 88, 100, 90, dan 70 adalah nilai masing-masing elemen; sedangkan `$A[0]`, `$A[1]`, `$A[2]`, `$A[3]`, dan `$A[4]` merupakan cara yang digunakan untuk melakukan pengaksesan setiap elemen *array*. Angka 0 sampai 4 adalah indeks dari *array* \$A. Ingat! Indeks *array* dalam PHP selalu dimulai dari 0, bukan 1. Ukuran dari *array* di atas dihitung berdasarkan jumlah elemen yang ada. Ukuran *array* dapat diambil menggunakan fungsi `sizeof()`.

```
$ukuran = sizeof($A);
```

Melalui kode di atas, variabel \$ukuran akan bernilai 5.

Array Asosiatif

Array asosiatif adalah *array* yang diindeks berdasarkan nama tertentu. Sebenarnya perbedaan antara *array* berindeks dan *array* asosiatif hanya terletak pada penamaan indeksnya saja.

101	88	100	90	70
<code>\$A["nol"]</code>	<code>\$A["satu"]</code>	<code>\$A["dua"]</code>	<code>\$A["tiga"]</code>	<code>\$A["empat"]</code>

Seperti yang Anda lihat pada gambar di atas, kali ini, *array* \$A diindeks berdasarkan nama, bukan berdasarkan nomor. Nama indeks yang digunakan di atas bertipe *string*.

Membuat Array

Untuk membuat *array*, Anda hanya perlu memasukkan nilai ke setiap elemennya. Sebagai contoh, jika Anda ingin membuat *array* dengan nama \$mobil, maka Anda dapat menulis kode berikut:

```
$mobil[0] = "Toyota";
$mobil[1] = "Honda";
$mobil[2] = "Suzuki";
$mobil[3] = "Daihatsu";
$mobil[4] = "Chevrolet";
$mobil[5] = "Mitsubishi";
```

Dalam PHP, Anda dapat menambah jumlah elemen *array* dengan mudah. Anda hanya perlu menambahkan nilai yang diinginkan ke dalamnya, misalnya seperti berikut:

```
$mobil[6] = "Ford";
$mobil[7] = "BMW";
```

PHP juga menyediakan menyediakan fungsi khusus untuk membuat *array*, yaitu `array()`.

- Untuk *array* berindeks:

```
$suku = array("Jawa", "Sunda", "Batak",  
            "Minang", "Dayak");
```

- Untuk *array* asosiatif:

```
$pegawai = array("dirut" => "Fredri",  
                "manager" => "Santoso",  
                "marketing" => "Shinta");
```

Mengakses Elemen Array

Setelah melakukan pembuatan dan pengisian nilai terhadap suatu *array*, langkah selanjutnya yang pasti akan Anda lakukan adalah mengakses nilai-nilai yang terkandung di dalamnya untuk memenuhi kebutuhan logika dari program yang Anda buat. Cara mengakses elemen *array* sangatlah sederhana. Anda dapat menempatkan nilai yang diakses ke dalam suatu variabel atau dapat juga langsung diproses dalam perhitungan maupun langsung ditampilkan.

Contoh:

```
$asal = $suku[0];  
echo $asal;
```

Melalui kode di atas, variabel `$asal` akan diisi dengan nilai dari elemen pertama array `$suku`. Dengan demikian, variabel `$asal` akan berisi teks "Jawa".

Cara mengakses *array* asosiatif pun sama, Anda hanya perlu menuliskan kunci (*key*) dari elemen yang akan diakses.

Contoh:

```
$direktur = $pegawai["dirut"];  
echo "Direktur perusahaan: $direktur";
```

Dengan kode tersebut, variabel `$direktur` akan bernilai "Fredri".

Praktikum 8-1: Membuat dan Mengakses Elemen Array (Bagian I)

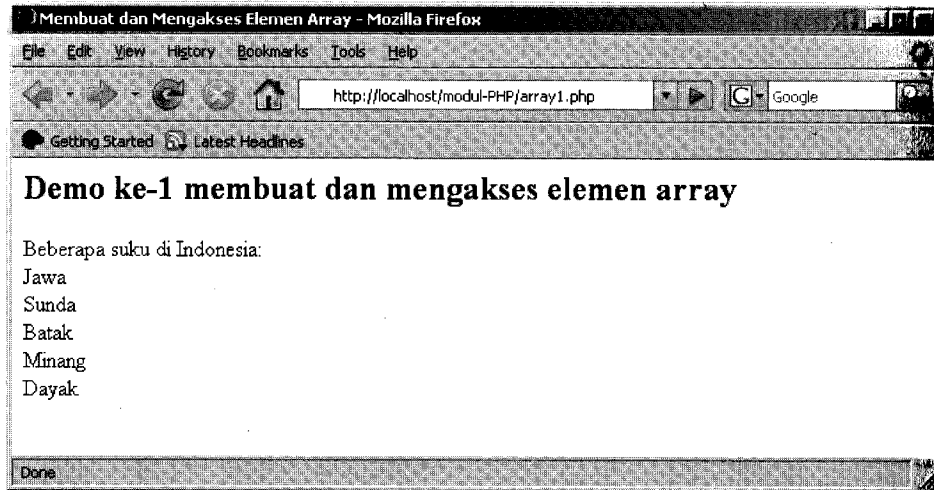
1. Jalankan aplikasi Text Editor

2. Tuliskan kode berikut:

```
<html>  
<head>  
    <title>Membuat dan Mengakses Elemen Array</title>  
</head>  
<body>  
  
    <?php  
  
    function judul() {  
        echo "<h2>Demo ke-1 membuat "  
            "dan mengakses elemen array</h2>";  
    }  
  
    judul();  
  
    $suku[0] = "Jawa";  
    $suku[1] = "Sunda";  
    $suku[2] = "Batak";  
    $suku[3] = "Minang";  
    $suku[4] = "Dayak";  
  
    echo "Beberapa suku di Indonesia: <br />";  
    echo $suku[0] . "<br />";  
    echo $suku[1] . "<br />";  
    echo $suku[2] . "<br />";  
    echo $suku[3] . "<br />";  
    echo $suku[4] . "<br />";  
  
    ?>
```

```
</body>
</html>
```

- ☑ Simpan file dengan nama **array1.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/array1.php**



Keterangan:

Pada kode di atas, mula-mula kita membuat dan mengisi nilai ke dalam array `$suku`. Kode yang digunakan untuk keperluan itu adalah:

```
$suku[0] = "Jawa";
$suku[1] = "Sunda";
$suku[2] = "Batak";
$suku[3] = "Minang";
$suku[4] = "Dayak";
```

Jika ingin lebih sederhana, Anda dapat mengganti kode tersebut dengan kode berikut:

```
$suku = array("Jawa", "Sunda", "Batak",
             "Minang", "Dayak");
```

Setelah array terbentuk, langkah selanjutnya yang kita lakukan adalah mengakses semua nilai yang terkandung di dalamnya, melalui kode berikut:

```
echo $suku[0] . "<br />";
echo $suku[1] . "<br />";
echo $suku[2] . "<br />";
echo $suku[3] . "<br />";
echo $suku[4] . "<br />";
```

Praktikum 8-2: Membuat dan Mengakses Elemen Array (Bagian 2)

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>Membuat dan Mengakses Elemen Array</title>
</head>
<body>

<?php

function judul() {
    echo "<h2>Demo ke-2 membuat " .
        "dan mengakses elemen array</h2>";
}

judul();

$pegawai = array("dirut" => "Fredri",
                 "manager" => "Santoso",
                 "marketing" => "Shinta");

echo "Pegawai perusahaan XYZ: <br />";
echo "<strong>Dirut:</strong> " .
    $pegawai["dirut"] . "<br />";
echo "<strong>Manager:</strong> " .
    $pegawai["manager"] . "<br />";
echo "<strong>Marketing:</strong> " .
    $pegawai["marketing"] . "<br />";

?>
```

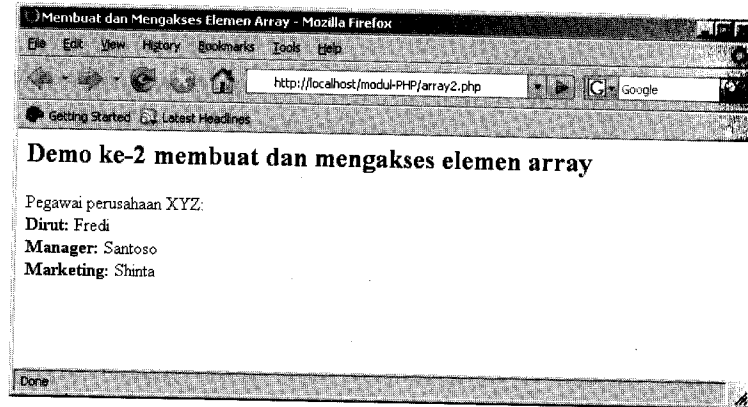
```
</body>
</html>
```

- ☑ Simpan *file* dengan nama **array2.php**
- ☑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/array2.php**

Keterangan:

Pada praktikum kali ini, kita mempraktikkan cara mengakses elemen-elemen yang terdapat pada array asosiatif.

Seperti yang Anda lihat bahwa dengan menyebutkan kunci dari masing-masing elemen, kita dapat dengan mudah mengakses nilai elemen yang kita inginkan.



Menggunakan Pengulangan untuk Mengakses Elemen Array

Pada praktikum di atas, Anda telah melakukan pengaksesan terhadap elemen *array* dengan cara mengambilnya satu per satu. Sebenarnya cara tersebut tidak efisien dan tidak direkomendasikan. Dalam praktik di lapangan kerja, pengaksesan *array* pada umumnya dilakukan dengan menggunakan struktur pengulangan. Dengan demikian, kode program yang kita tulis bisa jauh lebih sederhana dibandingkan dengan kode di atas.

Anda bisa menggunakan jenis pengulangan apa saja (*while*, *do-while*, maupun *for*) untuk mengakses elemen *array*, tergantung dari selera Anda sendiri. Namun, untuk jenis *array* asosiatif, biasanya jenis pengulangan yang paling lazim digunakan adalah *foreach*.

Contoh:

```
for ($i=0; $i < sizeof($suku); $i++) {
    echo $suku[$i] . "<br />";
}

foreach ($pegawai as $jabatan => $nama) {
    echo "$jabatan: $nama" . "<br />";
}
```

Praktikum 8-3: Menggunakan pengulangan untuk mengakses elemen *array*

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>Menggunakan Pengulangan
    untuk Mengakses Elemen Array</title>
</head>
<body>

<?php

function judul() {
    echo "<h2>Demo menggunakan pengulangan " .
        "untuk mengakses elemen array</h2>";
}

judul();

$suku = array("Jawa", "Sunda", "Batak",
    "Minang", "Dayak");

echo "Beberapa suku di Indonesia:<br />";
```



```
for ($i=0; $i < sizeof($suku); $i++) {
    echo $suku[$i] . "<br />";
}
```

```
$pegawai = array("dirut" => "Fredri",
    "manager" => "Santoso",
    "marketing" => "Shinta");
```

```
echo "Pegawai perusahaan XYZ: <br />";
foreach ($pegawai as $jabatan => $nama) {
    echo "<strong>$jabatan:</strong> " .
        $nama . "<br />";
}
```

```
?>
</body>
</html>
```

Maksud dari kode di atas adalah mengakses semua elemen yang ada dalam array `$pegawai`. Pada setiap elemen yang sedang dibaca, kunci (*key*) atau nama indeks *array* akan ditampung ke dalam variabel `$jabatan`, sedangkan nilainya ditampung ke dalam variabel `$nama`.

Memodifikasi Array

Nilai elemen *array* dapat diubah sesuai kebutuhan program. Caranya sama saja seperti pada saat proses pengisian nilai ke dalam elemen *array*. Elemen *array* yang nilainya diisi ulang, nilai lamanya akan tertimpa, tidak ada bedanya dengan variabel.

Contoh:

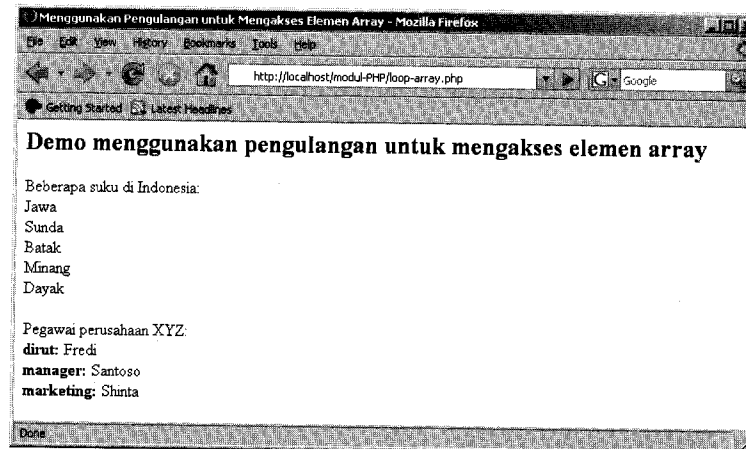
```
$A[0] = 10;
$A[1] = 20;
echo $A[0] . "<br />";
echo $A[1];
$A[0] = 100;
$A[1] = 200;
echo $A[0] . "<br />";
echo $A[1];
```

Pada contoh kode di atas, mula-mula *array* `$A` diisi dengan nilai 10 dan 20. Setelah itu, nilainya diganti menjadi 100 dan 200. Dengan demikian, pada saat elemen dari *array* tersebut dipanggil atau diakses, maka nilai yang akan diberikan adalah nilai terakhir, yaitu 100 dan 200.

Mengurutkan Array

PHP menyediakan beberapa fungsi untuk melakukan pengurutan terhadap elemen *array*. Fungsi mana yang harus digunakan tentunya harus disesuaikan dengan kebutuhan program yang sedang kita hadapi.

- ☑ Simpan *file* dengan nama **loop-array.php**
- ☑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/loop-array.php**



Keterangan:

Perhatikan
kode berikut:

```
foreach ($pegawai as $jabatan => $nama) {
    echo "<strong>$jabatan:</strong> " .
        $nama . "<br />";
}
```


Fungsi-fungsi untuk mengurutkan elemen *array* antara lain:

- `sort()`
- `asort()`
- `ksort()`
- `rsort()`
- `arsort()`
- `krsort()`

Menggunakan Fungsi `sort()`

Fungsi `sort()` akan mengurutkan *array* berdasarkan nilai elemennya. Dalam proses pengurutan ini, indeks *array* akan diset ulang. Jadi, yang dipindah/diurutkan hanya nilainya saja.

Praktikum 8-4: Mengurutkan array menggunakan fungsi `sort()`

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Mengurutkan Array</title>
</head>
<body>

<?php
function judul() {
  echo "<h2>Demo mengurutkan array " .
    "menggunakan fungsi sort()</h2>";
}

judul();

$suku = array("Jawa", "Sunda", "Batak",
  "Minang", "Dayak");

echo "<strong>Sebelum diurutkan:</strong><br />";
```

```
foreach ($suku as $key => $value) {
  echo "$key: $value" . "<br />";
}

sort($suku);

echo "<br /><strong>Setelah diurutkan:</strong><br />";
foreach ($suku as $key => $value) {
  echo "$key: $value" . "<br />";
}

?>
</body>
</html>
```

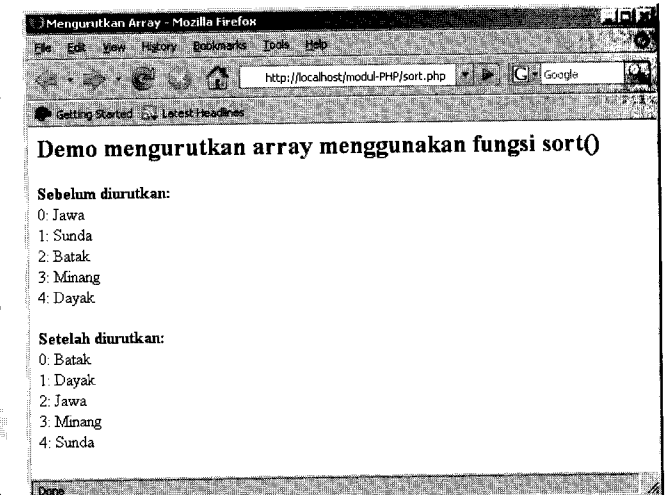
- ☑ Simpan file dengan nama **sort.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/sort.php**

Keterangan:

Seperti yang Anda lihat pada kode di atas, kita dapat dengan mudah mengurutkan semua elemen-elemen yang terdapat pada array `$suku`, dengan cara:

```
sort($suku);
```

Coba Anda perhatikan hasil yang diperoleh setelah proses pengurutan. Di situ tampak bahwa yang dipindah/diurutkan hanyalah nilainya saja, posisi indeks tetap seperti semula. Hal ini disebabkan karena dalam pemanggilan fungsi



`sort()`, indeks array akan diset ulang setelah elemen array diurutkan. Dengan demikian, hasil yang diperoleh adalah:

0: Batak
1: Dayak
2: Jawa
3: Minang
4: Sunda
bukan:

2: Batak
4: Dayak
0: Jawa
3: Minang
1: Sunda

Menggunakan Fungsi `asort()`

Untuk mengurutkan elemen-elemen pada array asosiatif, gunakan fungsi `asort()`. Dalam proses pengurutan ini, pasangan kunci/indeks dan nilai akan tetap dijaga, tidak seperti yang terjadi pada fungsi `sort()`.

Praktikum 8-5: Mengurutkan array menggunakan fungsi `asort()`

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Mengurutkan Array</title>
</head>
<body>

<?php

function judul() {
```

```
echo "<h2>Demo mengurutkan array " .
      "menggunakan fungsi asort()</h2>";
}

judul();

$suku = array('0' => "Jawa",
              '1' => "Sunda",
              '2' => "Batak",
              '3' => "Minang",
              '4' => "Dayak");

echo "<strong>Sebelum diurutkan:</strong><br />";
foreach ($suku as $key => $value) {
    echo "$key: $value" . "<br />";
}

asort($suku);

echo "<br /><strong>Setelah diurutkan:</strong><br />";
foreach ($suku as $key => $value) {
    echo "$key: $value" . "<br />";
}

?>
</body>
</html>
```

- ☒ Simpan file dengan nama **asort.php**
- ☒ Jalankan web browser dan masukkan **http://localhost/modul-PHP/asort.php**

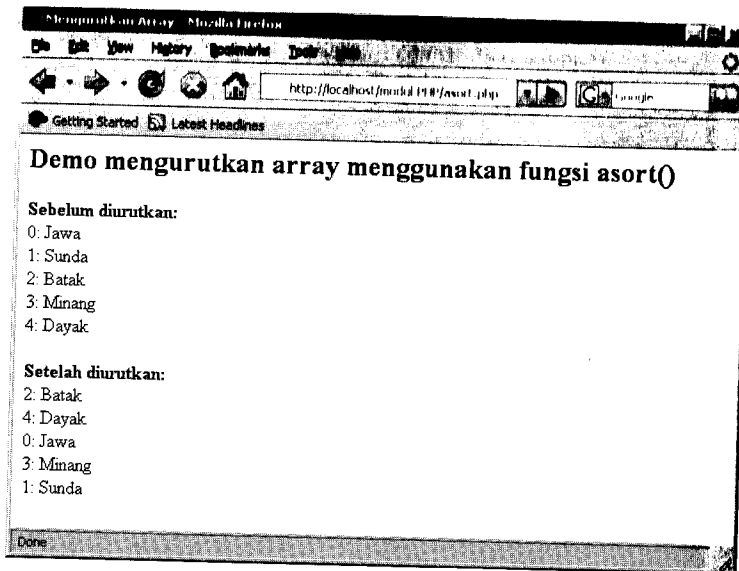
Keterangan:

Pada praktikum kali ini, kita membuat array \$suku menjadi array asosiatif, dengan cara menamai indeks array tersebut dengan string '1', '2', '3', '4', dan '5'. Pada hasil yang diperoleh di atas dapat

Anda lihat bahwa pasangan indeks dan nilai setiap elemen array tidak terpisah. Di sini, yang diurutkan hanya nilainya. Dalam kasus ini, nilai elemen array diurutkan berdasarkan urutan alfabet karena nilainya berupa string. Meskipun demikian, Anda tetap dapat menggunakan fungsi `asort()` untuk mengurutkan elemen array yang nilainya berupa numerik. Kesimpulannya, fungsi `asort()` tidak akan mempedulikan apakah indeks terurut atau tidak.

Menggunakan Fungsi `ksort()`

Dua fungsi yang telah kita bahas di atas melakukan pengurutan berdasarkan nilai elemen array. PHP juga menyediakan fungsi khusus untuk mengurutkan elemen array berdasarkan kunci (*key*) atau indeksinya, yaitu `ksort()`. Dengan kata lain, dalam pengurutan menggunakan `ksort()`, yang akan diurutkan hanya indeksnya saja. Sama halnya seperti `asort()`, pada `ksort()` pasangan kunci dan nilai juga akan tetap terjaga.



Praktikum 8-6: Mengurutkan array menggunakan fungsi `ksort()`

1. Jalankan aplikasi Text Editor

2. Tuliskan kode berikut:

```
<html>
<head>
    <title>Mengurutkan Array</title>
</head>
<body>

<?php

function judul() {
    echo "<h2>Demo mengurutkan array " .
        "menggunakan fungsi ksort()</h2>";
}

judul();

$suku = array('nol' => "Jawa",
    'satu' => "Sunda",
    'dua' => "Batak",
    'tiga' => "Minang",
    'empat' => "Dayak");

echo "<strong>Sebelum diurutkan:</strong><br />";
foreach ($suku as $key => $value) {
    echo "$key: $value" . "<br />";
}

ksort($suku);

echo "<br /><strong>Setelah diurutkan:</strong><br />";
foreach ($suku as $key => $value) {
    echo "$key: $value" . "<br />";
}

?>

</body>
</html>
```

- ☑ Simpan file dengan nama **ksort.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/ksort.php**

Keterangan:

Kali ini, kita mengganti nama indeks array dengan string 'nol', 'satu', 'dua', 'tiga', dan 'empat'. Nama indeks inilah yang akan diurutkan oleh fungsi `ksort()`, bukan nilai elemennya. Dengan demikian, hasil yang akan diperoleh setelah proses pengurutan adalah:

dua
empat
nol
satu
tiga

Nama indeks array di atas telah diurutkan berdasarkan urutan alfabet.

Menggunakan Fungsi `rsort()`, `arsort()`, dan `krsort()`

Fungsi `sort()`, `asort()`, dan `ksort()` yang telah dibahas di atas digunakan untuk melakukan pengurutan array secara menaik (*ascending*). Untuk melakukan pengurutan terbalik/menurun (*descending*), PHP menyediakan fungsi `rsort()`, `arsort()`, dan `krsort()`. Fungsi

`rsort()` memiliki konsep kerja yang sama seperti fungsi `sort()`, perbedaannya hanya data akan diurut secara terbalik (data dengan nilai paling tinggi akan disimpan pada urutan pertama). Begitu juga dengan `arsort()`, cara kerjanya sama dengan `asort()`; sedangkan `krsort()`, cara kerjanya sama dengan `ksort()`.

Praktikum 8-8: Mengurutkan array secara *descending*

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>Mengurutkan Array</title>
</head>
<body>

<?php

function judul() {
    echo "<h2>Demo mengurutkan array " .
        "secara descending</h2>";
}

judul();

$suku = array('nol' => "Jawa",
              'satu' => "Sunda",
              'dua' => "Batak",
              'tiga' => "Minang",
              'empat' => "Dayak");

echo "<strong>Sebelum diurutkan:</strong><br />";
foreach ($suku as $key => $value) {
    echo "$key: $value" . "<br />";
}

rsort($suku);

echo "<br /><strong>Setelah diurutkan:</strong><br />";
foreach ($suku as $key => $value) {
    echo "$key: $value" . "<br />";
}
```

)

?>

</body>

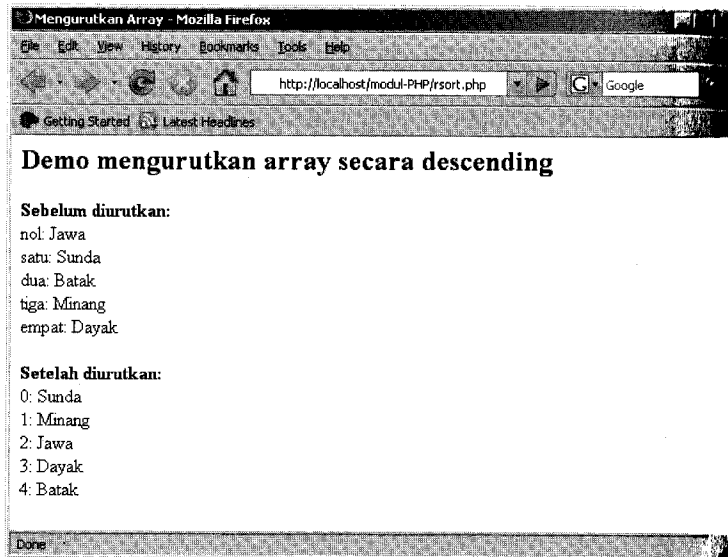
</html>

- Simpan file dengan nama **rsort.php**
- Jalankan web browser dan masukkan **http://localhost/modul-PHP/rsort.php**

Keterangan:

Pada praktikum ini Anda bisa lihat bahwa dengan menggunakan fungsi `rsort()`, array `$suku` akan diurutkan secara terbalik (descending).

Karena `rsort()` memiliki konsep yang sama dengan `sort()`, maka indeks array hasil pengurutan akan direset ulang, dan semuanya berupa numerik, bukan berupa nama/string. Anda dapat memodifikasi program di atas dengan mengganti fungsi `sort()` menjadi `arsort()` maupun `krsort()`, lalu perhatikan perbedaan yang akan diperoleh.



Membuat Form

Form web merupakan suatu antarmuka (*user interface*) agar user dapat berkomunikasi dengan aplikasi. Setiap permintaan yang diminta oleh user (disebut *request*) akan direspon oleh aplikasi, kemudian hasilnya akan dikembalikan lagi ke hadapan user melalui web browser. Dalam modul ini Anda akan mempelajari materi-materi berikut:

- Mengenal tag HTML `<form>...</form>`
- Mengenal metode pengiriman GET dan POST
- Memproses data yang dikirim dari klien
- Membuat dan memproses beragam elemen kontrol di dalam form

Mengenal Tag `<form>..</form>`

Untuk membuat *form* dalam halaman *web*, kita perlu menggunakan *tag* HTML `<form>` dan `</form>`.

Dalam bekerja dengan *form*, Anda pasti akan menggunakan tombol `SUBMIT` yang digunakan untuk mengirimkan semua data yang ditulis oleh *user* melalui *form* ke *server*. Setiap *form* yang ditampilkan pada halaman *web* pasti akan memiliki tombol `SUBMIT`. Maka dari itu, tombol ini memiliki peranan yang sangat penting dalam pembuatan *form web*.

Ketika *user* mengisi data pada *form* yang tersedia, ada kalanya *user* salah dalam melakukan hal tersebut sehingga ingin mengembalikan data ke nilai semula (*default*). Untuk kebutuhan ini, HTML menyediakan tombol `RESET`.

Selain itu, dalam *tag* `<form>`, terdapat dua atribut penting yang perlu Anda ketahui kegunaannya, yaitu:

- `ACTION`
- `METHOD`

Selain dua atribut tersebut, *tag* `<form>` juga memiliki banyak atribut lain, seperti `ID`, `LANG`, `DIR`, `CLASS`, `LANGUAGE`, `NAME`, `STYLE`, dan `TITLE`. Atribut-atribut ini bersifat umum sama seperti pada elemen kontrol HTML lainnya sehingga kita tidak akan membahasnya lebih detail.

Atribut `ACTION`

Atribut `ACTION` digunakan untuk menunjuk *file* PHP yang akan digunakan untuk memproses data yang dikirim melalui *form*. *File* tersebut akan dieksekusi ketika pengguna *web* memilih tombol `SUBMIT`.

Contoh:

```
<form action="contoh.php">
...
</form>
```

Pada contoh kode di atas, *file* yang akan dipanggil dan dieksekusi pada saat *user* memilih tombol `SUBMIT` adalah **contoh.php**.

Atribut `METHOD`

Atribut `METHOD` digunakan untuk menentukan bagaimana cara informasi dari suatu *form* dikirimkan ke *server*. Kita dapat menggunakan salah satu dari dua cara: menggunakan *method* `GET` atau `POST`.

```
<form action="contoh.php" method="get">
...
</form>
```

atau

```
<form action="contoh.php" method="post">
...
</form>
```

Catatan:

Sebenarnya terdapat nilai atribut `METHOD` yang lain seperti `HEAD`, `PUT`, `LINK`, `UNLINK`, `OPTIONS`, `DELETE`, `TRACE`, dan `CONNECT`. Namun, karena nilai-nilai tersebut tidak umum digunakan dalam pembuatan *form* maka kita tidak akan membahasnya.

Menggunakan Metode `GET`

Nilai `GET` pada atribut `METHOD` digunakan untuk mengirimkan data ke *server* dengan memerintahkan *web browser* untuk menambah nilai yang dimasukkan oleh *user* ke dalam URL. Informasi yang akan ditambahkan ke URL memiliki format berupa **pasangan nama/nilai**, yang ditulis

nama=nilai. **nama** yang dimaksud di sini adalah nama dari elemen kontrol yang menyimpan data (misalnya elemen *Text*), sedangkan **nilai** adalah data yang dimasukkan oleh *user*.

Contoh:

?pekerjaan=Dosen

Di sini, “pekerjaan” adalah nama elemen kontrol dan “Dosen” adalah nilainya.

Jika data yang dikirimkan lebih dari satu, maka akan ditambahkan tanda &.

Contoh:

?nama=Mira&gender=wanita&pekerjaan=Dosen

Praktikum 9-1: Mengirim data menggunakan metode GET

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Metode GET</title>
</head>
<body>

<h2>Demo mengirim data menggunakan GET</h2>

<form action="get.php" method="get">
  Masukkan nama Anda:
  <input type="text" name="nama" size="20">
  <input type="submit" value="Kirim">
</form>

</body>
</html>
```

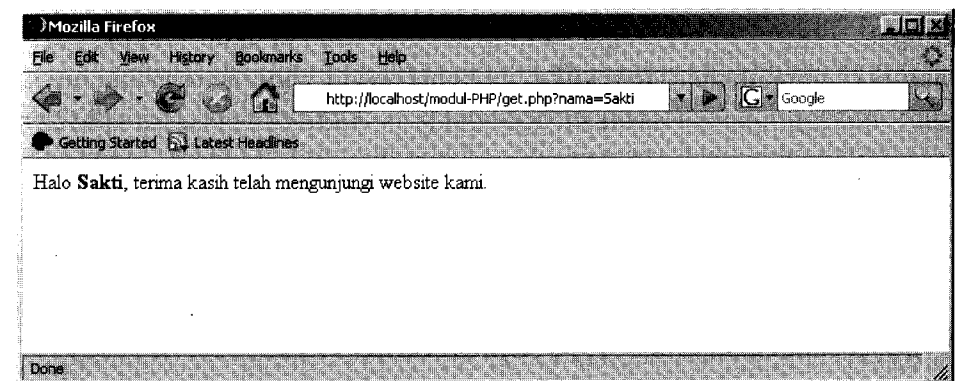
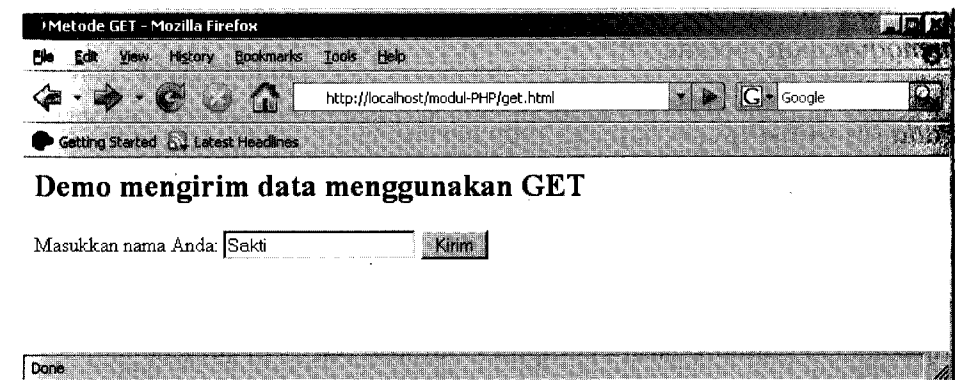
☑ Simpan *file* dengan nama **get.html**

☑ Buatlah *file* baru, dan tuliskan kode berikut:

```
<?php
  $nama = $_GET["nama"];
  if (isset($nama)) {
    echo "Halo <strong>$nama</strong>, " .
      "terima kasih telah mengunjungi website kami.";
  }
?>
```

☑ Simpan *file* dengan nama **get.php**

☑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/get.html**



Keterangan:

Perhatikan file **get.html**. Di situ tampak bahwa atribut `ACTION` diisi dengan **get.php**.

```
action = "get.php"
```

Ini berarti bahwa file yang akan diproses ketika user memilih tombol `SUBMIT` adalah file **get.php**.

Selanjutnya, atribut `METHOD` diisi dengan "get".

```
method = "get"
```

Ini berarti bahwa pengiriman data ke server akan dilakukan menggunakan metode `GET`.

Pada file **get.php**, perhatikan kode berikut:

```
$nama = $_GET["nama"];
```

Kode di atas digunakan untuk mengambil nilai dari elemen kontrol yang bernama **nama**, yang dikirimkan menggunakan method `GET`. Nilai yang diperoleh kemudian akan disimpan ke dalam variabel **\$nama**.

Fungsi `isset()` berfungsi untuk memeriksa apakah suatu variabel memiliki nilai atau tidak. Jika variabel yang diperiksa tidak memiliki nilai maka fungsi ini akan mengembalikan nilai `false`. Jika sebaliknya, maka nilai yang akan dikembalikan adalah `true`.

Coba Anda perhatikan URL yang muncul pada address bar dalam web browser yang Anda gunakan:

<http://localhost/modul-PHP/get.php?nama=Sakti>

Tampak bahwa nama elemen kontrol dan nilainya akan ditampilkan di dalam URL. Tanda `?` digunakan sebagai pembatas antara URL dengan parameter-parameter yang dilewatkan oleh form.

Menggunakan Metode `POST`

Nilai `POST` pada atribut `METHOD` dari tag `<form>` pada dasarnya memiliki fungsi yang sama seperti `GET`. Perbedaanya, pada metode pengiriman ini, data-data yang dimasukkan oleh user tidak akan ditambahkan/ditampilkan ke dalam URL.

Praktikum 9-2: Mengirim data menggunakan metode `POST`

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
    <title>Metode POST</title>
</head>
<body>

<h2>Demo mengirim data menggunakan POST</h2>
<form action="post.php" method="post">
    Masukkan nama Anda:
    <input type="text" name="nama" size="20">
    <input type="submit" value="Kirim">
</form>

</body>
</html>
```

- ☒ Simpan file dengan nama **post.html**
- ☒ Buatlah file baru, dan tuliskan kode berikut:

```
<?php
    $nama = $_POST["nama"];
    if (isset($nama)) {
        echo "Halo <strong>$nama</strong>, " .
            "terima kasih telah mengunjungi website kami.";
    }
```

- ☑ Simpan file dengan nama **post.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/post.html**

Keterangan:
Kali ini, dalam file **post.html** kita menggunakan metode **POST** dalam pengiriman data dari klien ke server.

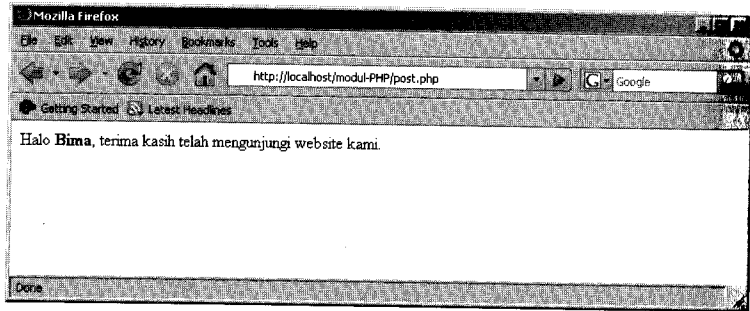
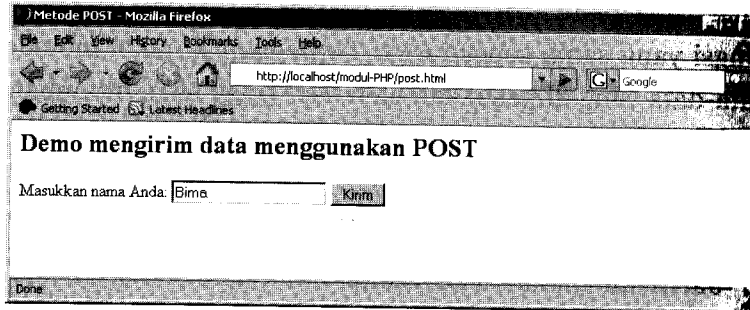
method = "post"

Maka dari itu, cara pengambilan nilai variabelnya pun harus berbeda dengan sebelumnya, yaitu:

```
$nama = $_POST["nama"];
```

Selain itu, coba Anda perhatikan juga URL yang dihasilkan:

http://localhost/modul-PHP/post.php



Tampak pada URL tersebut bahwa data yang dikirim melalui form tidak ikut ditampilkan di dalam URL. Kesimpulannya, data yang dikirim menggunakan metode **POST** akan disembunyikan.

Membuat Elemen Kontrol di dalam Form

Form yang dijumpai dalam suatu web biasanya terdiri atas beragam jenis elemen kontrol: seperti *text field*, *radio button*, *check box*, *list (combo box maupun list box)*, dan lain-lain. Elemen-elemen kontrol tersebut pada umumnya dibuat menggunakan tag `<input>`, meskipun ada juga yang dibuat menggunakan tag `<select>`. Pada bagian ini, kita akan mempelajari bagaimana PHP menangani masing-masing elemen kontrol tersebut.

Elemen TEXT FIELD

Text field (sering juga disebut *text box*) merupakan elemen kontrol yang paling banyak dijumpai dalam suatu form, seperti pada pengisian *keyword* untuk pencarian, pengisian nama, alamat *e-mail*, dan sebagainya. *Text field* hanya dapat menampung satu baris teks. Jika teks yang kita inginkan berjumlah lebih dari satu baris, maka kita perlu menggunakan elemen *text area*.

Praktikum 9-3: Menggunakan elemen TEXT FIELD

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Elemen Text Field</title>
</head>
<body>

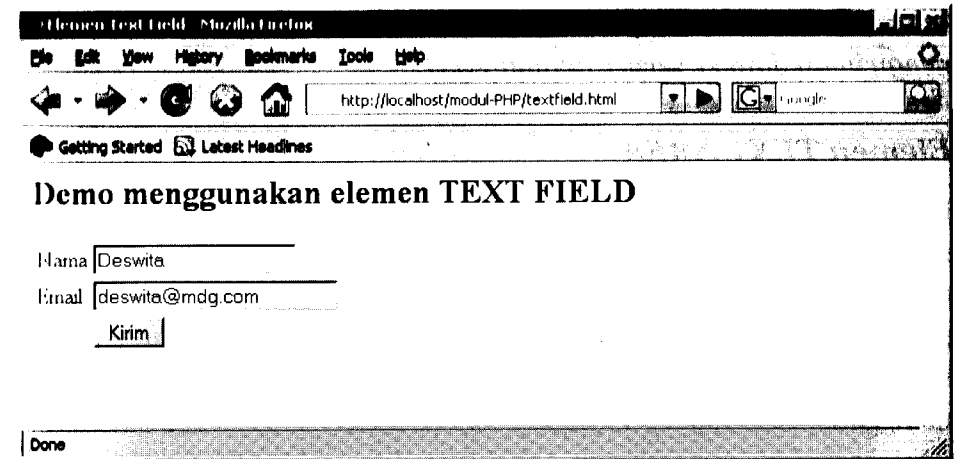
  <h2>Demo menggunakan elemen TEXT FIELD</h2>
```

```
<form action="textfield.php" method="post">
  <table border="0">
    <tr>
      <td>Nama</td>
      <td><input type="text" name="nama" size="20"
        maxlength="25" /></td>
    </tr>
    <tr>
      <td>Email</td>
      <td><input type="text" name="email" size="25"
        maxlength="35" /></td>
    </tr>
    <tr>
      <td></td>
      <td><input type="submit" name="btnSubmit"
        value="Kirim" /></td>
    </tr>
  </table>
</form>
</body>
</html>
```

- ☑ Simpan file dengan nama **textfield.html**
- ☑ Buatlah file baru, dan tuliskan kode berikut:

```
<?php
if (isset($_POST["btnSubmit"])) {
    $nama = $_POST["nama"];
    $email = $_POST["email"];
    if (isset($nama) && isset($email)) {
        echo "<h3>Data yang dimasukkan:</h3>";
        echo "Nama: $nama<br />";
        echo "Email: $email";
    }
}
?>
```

- ☑ Simpan file dengan nama **textfield.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/textfield.html**



Keterangan:

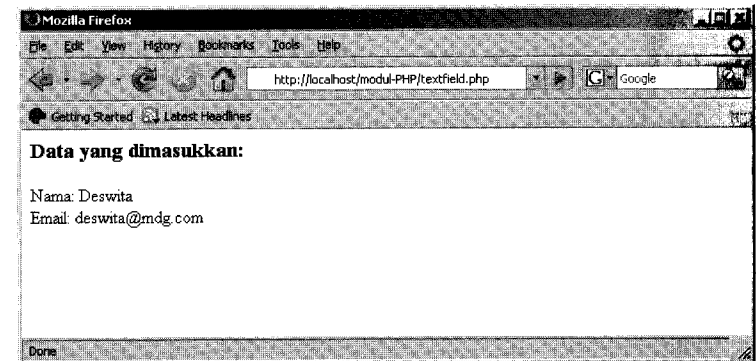
Dalam file **textfield.html**, mula-mula kita memeriksa apakah tombol **SUBMIT** telah

diklik atau belum. Kode yang digunakan adalah:

```
if (isset($_POST["btnSubmit"])) {
    ...
}
```

Jika ya, maka nilai-nilai yang terdapat pada elemen text field akan ditangkap dan ditampung ke dalam variabel **\$nama** dan **\$email**. Kode yang digunakan untuk keperluan ini adalah:

```
$nama = $_POST["nama"];
$email = $_POST["email"];
```



Kita menggunakan `$ POST` karena metode yang digunakan untuk pengiriman data ke server adalah `POST`.

Elemen TEXT AREA

Elemen *text area* memiliki fungsi yang sama seperti *text field*, yaitu untuk memasukkan data teks. Perbedaannya, *text area* mampu menampung teks yang jumlahnya lebih dari satu baris. Elemen *text area* biasa digunakan sebagai kontrol untuk data keterangan, alamat, isi pesan *e-mail*, dan sebagainya. Berbeda dengan elemen kontrol lain, *text area* tidak dibuat menggunakan tag `<input>`, melainkan tag `<textarea>` dan `</textarea>`.

Praktikum 9-4: Menggunakan elemen TEXT AREA

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

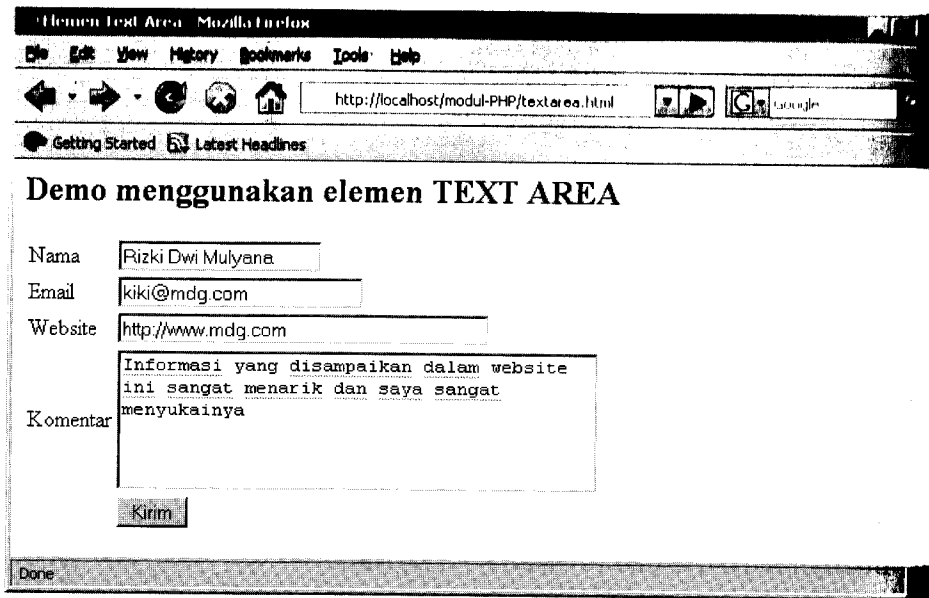
```
<html>
<head>
  <title>Elemen Text Area</title>
</head>
<body>
<h2>Demo menggunakan elemen TEXT AREA</h2>
<form action="textarea.php" method="post">
  <table border="0">
    <tr>
      <td>Nama</td>
      <td><input type="text" name="nama" size="20"
        maxlength="25" /></td>
    </tr>
    <tr>
      <td>Email</td>
      <td><input type="text" name="email" size="25"
        maxlength="35" /></td>
    </tr>
    <tr>
      <td>Website</td>
      <td><input type="text" name="website" size="40"
        maxlength="50" /></td>
```

```
</tr>
<tr>
  <td>Komentar</td>
  <td><textarea name="komentar"
    cols="40" rows="5"></textarea></td>
</tr>
<tr>
  <td></td>
  <td><input type="submit" name="btnSubmit"
    value="Kirim" /></td>
</tr>
</table>
</form>
</body>
</html>
```

- ✓ Simpan file dengan nama **textarea.html**
- ✓ Buatlah file baru, dan tuliskan kode berikut:

```
<?php
if (isset($_POST["btnSubmit"])) {
  $nama = $_POST["nama"];
  $email = $_POST["email"];
  $website = $_POST["website"];
  $komentar = $_POST["komentar"];
  echo "<h3>Data yang dimasukkan:</h3>";
  echo "<strong>Nama:</strong> $nama<br />";
  echo "<strong>Email:</strong> $email<br>";
  echo "<strong>Website:</strong> $website<br>";
  echo "<strong>Komentar:</strong> $komentar<br>";
}
?>
```

- ✓ Simpan file dengan nama **textarea.php**
- ✓ Jalankan web browser dan masukkan **http://localhost/modul-PHP/textarea.html**



Keterangan:

Pada praktikum di atas kita membuat kolom komentar dengan elemen kontrol text area karena komentar yang user masukkan bisa saja lebih dari satu baris. Kode yang kita gunakan dalam contoh ini adalah:

```
<textarea name="komentar" cols="40" rows="5"></textarea>
```

Atribut COLS menunjukkan jumlah kolom dan atribut ROWS menunjukkan jumlah baris yang ditampilkan.

Elemen CHECK BOX

Cara untuk memperoleh data yang dikirimkan melalui elemen *check box* relatif lebih rumit dibandingkan dengan elemen *text field* dan *text area*. Pasalnya, di sini ada kalanya kita perlu menggunakan *array* untuk menanganinya. *Check box* adalah elemen kontrol yang mengizinkan user untuk memilih lebih dari satu opsi, misalnya pada *form* yang meminta user memasukkan data hobi, di mana user dapat memilih lebih dari satu hobi.

Praktikum 9-5: Menggunakan elemen CHECK BOX

- M Jalankan aplikasi Text Editor
- M Tuliskan kode berikut:

```
<html>
<head>
  <title>Elemen Check Box</title>
</head>
<body>
<h2>Demo menggunakan elemen CHECK BOX</h2>
Apa hobi Anda?<br />
<form action="checkbox.php" method="post">
  <input type="checkbox" name="hobi[]"
    value="Olahraga">Olahraga<br />
  <input type="checkbox" name="hobi[]"
    value="Memancing">Memancing<br />
  <input type="checkbox" name="hobi[]"
    value="Membaca">Membaca<br />
  <input type="checkbox" name="hobi[]"
    value="Travelling">Travelling<br />
  <input type="submit" value="Kirim">
</form>
</body>
</html>
```

- ☑ Simpan file dengan nama **checkbox.html**
- ☑ Buatlah file baru, dan tuliskan kode berikut:

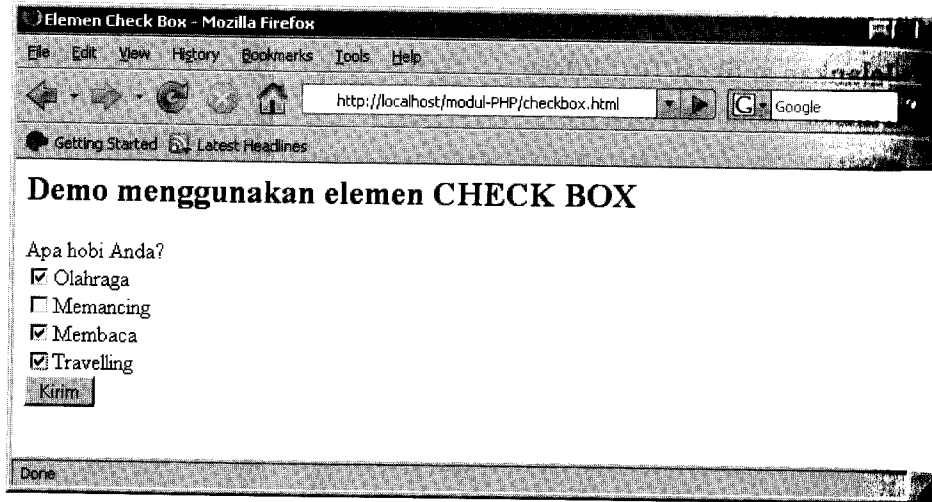
```
<?php
$hobi = $_POST["hobi"];
```



```
echo "<p>Hobi yang Anda pilih:</p>";
foreach ($hobi as $nilai) {
    echo "- $nilai <br />";
}
```

??

- ☑ Simpan file dengan nama **checkbox.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/checkbox.html**



Keterangan:

Dalam file **checkbox.html** kita membuat empat buah elemen **check box**. Keempat elemen ini kita jadikan sebagai array dengan memberi nama kontrol yang sama dan tanda bracket (**[]**) di belakangnya. Dengan

demikian, setiap elemen **check box** yang dipilih oleh user akan dimasukkan ke dalam array.

Karena berupa array, maka pemrosesannya juga berbeda. Pada praktikum kali ini, kita menggunakan statemen **foreach** untuk menampilkan isi array **\$hobi**.

```
foreach ($hobi as $nilai) {
    echo "- $nilai <br />";
}
```

Elemen **RADIO BUTTON**

Berbeda dengan elemen **check box** yang berfungsi untuk menampung beberapa pilihan, elemen **radio button** digunakan untuk menentukan satu pilihan dari beberapa opsi yang tersedia.

Praktikum 9-6: Menggunakan elemen **RADIO BUTTON**

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
    <title>Elemen Radio Button</title>
</head>
<body>
<h2>Demo menggunakan elemen RADIO BUTTON</h2>
<form action="radiobutton.php" method="post">
    Nama:
    <input type="text" name="nama" size="20" /> <br />
    Jenis kelamin:
    <input type="radio" name="gender"
        value="P" /> Pria
    <input type="radio" name="gender"
        value="W" /> Wanita<br />
    <p><input type="submit" value="Kirim" /></p>
</form>
</body>
</html>
```

- ☑ Simpan *file* dengan nama **radiobutton.html**
- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```
<?php
```

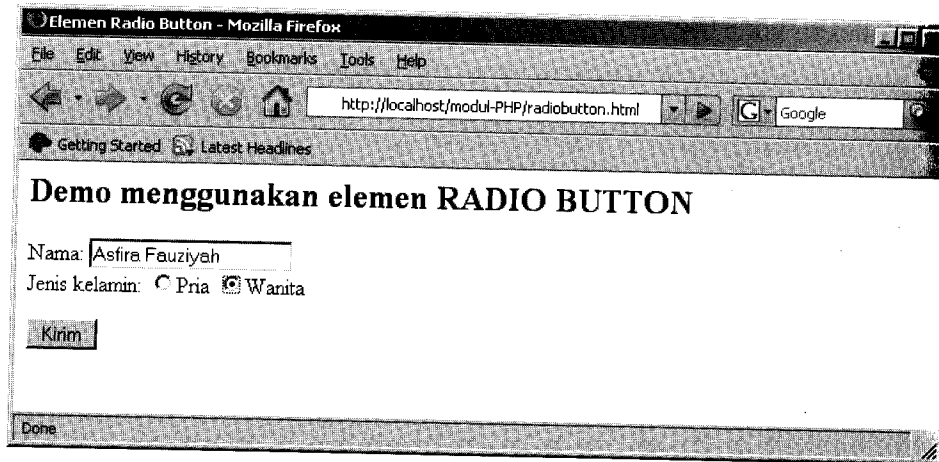
```
$nama = $_POST["nama"];
$gender = $_POST["gender"];

echo "<p>Data yang dimasukkan:</p>";
echo "<strong>Nama:</strong> $nama <br />";
echo "<strong>Jenis kelamin:</strong> ";

if ($gender == "P") {
    echo "Pria";
} else {
    echo "Wanita";
}

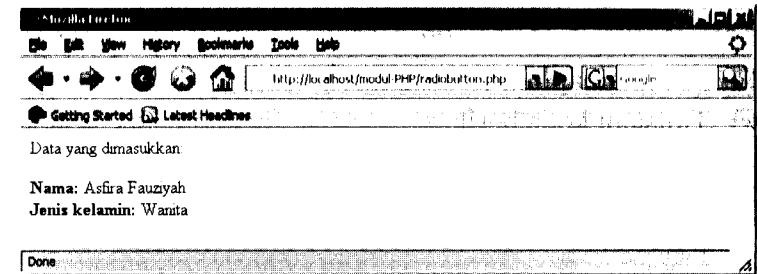
?>
```

- ☑ Simpan *file* dengan nama **radiobutton.php**
- ☑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/radiobutton.html**



Keterangan:

Pada praktikum kali ini, kita membuat dua buah opsi (Pria dan Wanita) menggunakan



elemen *radio button*. Ingat, nama elemen *radio button* harus sama, dan bukan berupa array (tidak diikuti tanda []).

- Input type="radio" name="gender" value="P" />Pria
- Input type="radio" name="gender" value="W" />Wanita

Dalam contoh ini, masing-masing elemen *radio button* kita namai dengan *gender*.

Elemen LIST

Dalam HTML, elemen *list* dibagi menjadi dua, yaitu sebagai *list box* dan *combo box*. *List* dibuat dengan menggunakan tag `<select>` dan `</select>`, sedangkan isi atau *item* dari *list* dibuat dengan menggunakan tag `<option>` dan `</option>`.

```
<select name="NamaKontrol" size="1">
    <option value="Nilai1">Item ke-1</option>
    <option value="Nilai2">Item ke-2</option>
    <option value="Nilai3">Item ke-3</option>
    ...
</select>
```

Jika kita ingin membuat elemen *list box* maka kita perlu menentukan atribut `SIZE` pada tag `<select>` dengan nilai lebih besar dari 1; sedangkan untuk elemen *combo box*, kita perlu mengisi `SIZE` dengan nilai 1.

Praktikum 9-7: Menggunakan elemen *LIST*

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Elemen List</title>
</head>
<body>
<h2>Demo menggunakan elemen list
  (list box dan combo box)</h2>
<form action="list.php" method="post">
  <p><strong>List Box</strong></p>
  Kota:<br/>
  <select name="kota" size="10">
    <option value="Jakarta">Jakarta</option>
    <option value="Depok">Depok</option>
    <option value="Bogor">Bogor</option>
    <option value="Sukabumi">Sukabumi</option>
    <option value="Cianjur">Cianjur</option>
    <option value="Cimahi">Cimahi</option>
    <option value="Bandung">Bandung</option>
  </select>
  <p><strong>Combo Box</strong></p>
  Agama:
  <select name="agama" size="1">
    <option value="Islam">Islam</option>
    <option value="Katolik">Katolik</option>
    <option value="Protestan">Protestan</option>
    <option value="Hindu">Hindu</option>
    <option value="Budha">Budha</option>
  </select>
  <p><input type="submit" value="Kirim" /></p>
</form>
</body>
</html>
```

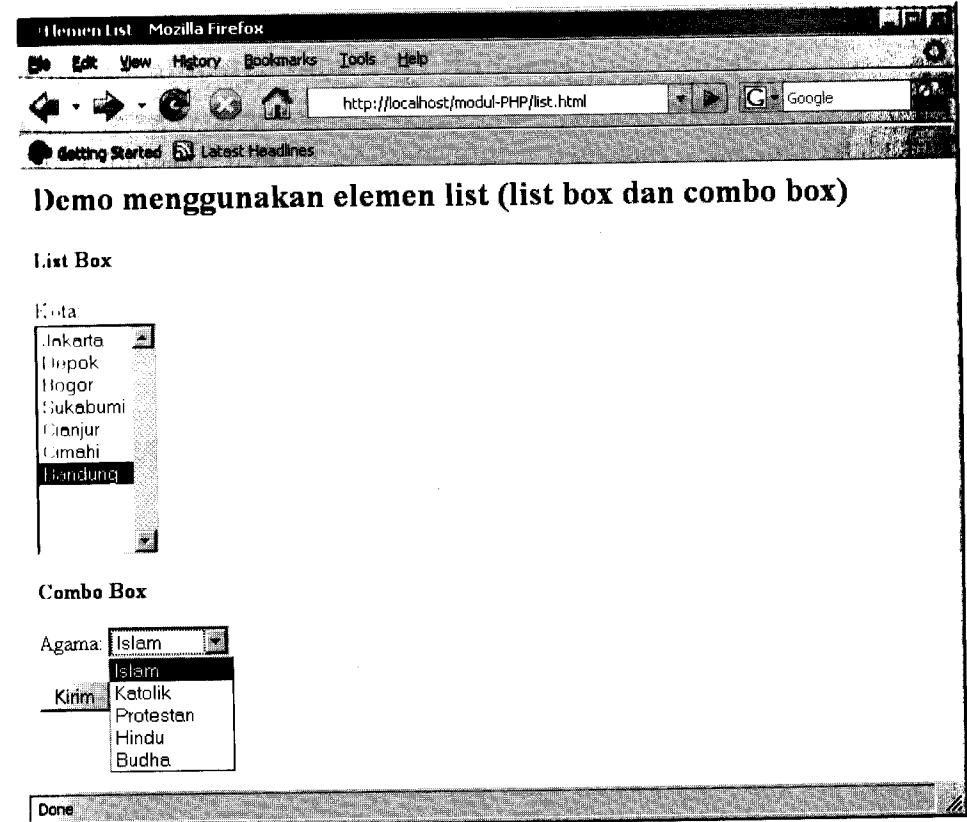
- ☑ Simpan *file* dengan nama **list.html**
- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```
<?php
$kota = $_POST["kota"];
```

```
$agama = $_POST["agama"];
```

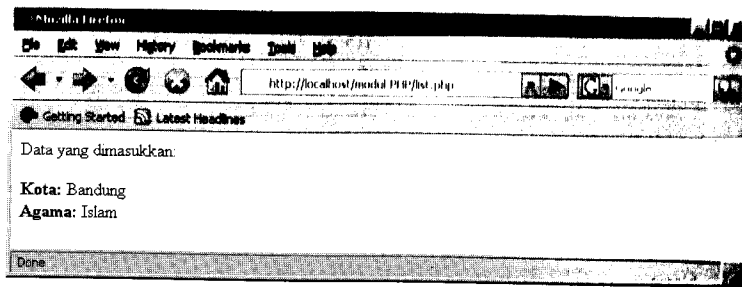
```
echo "<p>Data yang dimasukkan:</p>";
echo "<strong>Kota:</strong> $kota <br />";
echo "<strong>Agama:</strong> $agama ";
```

- 1. Simpan *file* dengan nama **list.php**
- 2. Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/list.html**



Keterangan:

Seperti yang Anda lihat di atas, perbedaan pembuatan list box dan combo box hanya



terletak pada nilai yang dimasukkan ke dalam atribut `SIZE`. Adapun cara menangkap nilai yang dikirim melalui kedua elemen list tersebut adalah sama seperti pada elemen kontrol lainnya.

Elemen HIDDEN CONTROL

Terkadang kita ingin mengambil informasi dari suatu halaman web, dan melewatkannya ke halaman lain tanpa harus meminta input dari user. Pada kasus seperti ini, kita perlu suatu elemen kontrol yang dapat disembunyikan. Elemen kontrol seperti ini disebut *hidden control*. Dalam HTML, kita dapat membuat kontrol seperti ini menggunakan tag `<INPUT>` dengan atribut `TYPE` diisi dengan nilai `HIDDEN`.

Praktikum 9-8: Menggunakan elemen HIDDEN CONTROL

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<html>
<head>
  <title>Elemen Hidden Control</title>
</head>
<body>
  <h2>Demo menggunakan elemen HIDDEN CONTROL</h2>
  Klik tombol di bawah ini!
  <form action="hiddencontrol.php" method="post">
    <input name="hc" type="hidden" value="100">
    <p><input type="submit" value="Kirim" /></p>
  </form>
</body>
</html>
```

1. Simpan file dengan nama **hiddencontrol.html**

2. Buatlah file baru, dan tuliskan kode berikut:

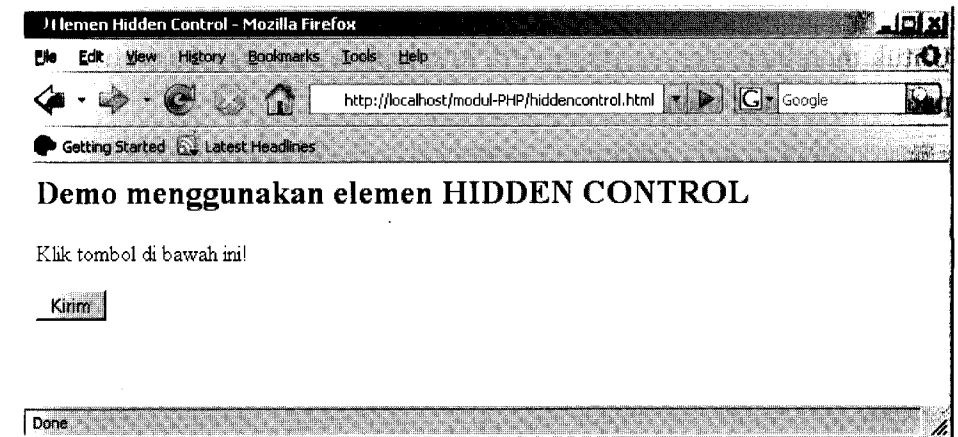
```
<?php
$hcvalue = $_POST["hc"];

echo "<p>Data yang dikirimkan " .
      "secara tersembunyi:</p>";
echo "$hcvalue";
?>
```

3. Simpan file dengan nama **hiddencontrol.php**

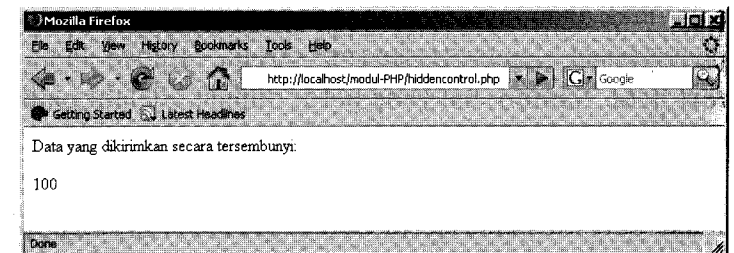
4. Jalankan web browser dan masukkan

http://localhost/modul-PHP/hiddencontrol.html



Keterangan:

Pada praktikum kali ini kita melewati nilai ke server dengan cara tersembunyi,



yaitu dengan menggunakan elemen *hidden*. Dalam contoh ini, nilai yang kita kirimkan adalah 100, yaitu yang kita isikan pada atribut *VALUE* pada tag `<input>`.

```
<input name="hc" type="hidden" value="100">
```

Elemen *PASSWORD*

Elemen *password* digunakan untuk melakukan input data yang bersifat rahasia. Perilaku elemen ini sebenarnya hampir sama dengan elemen *text field*. Perbedaanya, pada saat *user* mengisi data pada elemen *password*, teks yang diketikkan akan ditampilkan dengan tanda *asterisk* (*) maupun tanda bulatan kecil.

Untuk membuat elemen *password*, kita hanya perlu mengisi atribut *TYPE* pada tag `<input>` dengan nilai *PASSWORD*.

```
<input name="password" type="password">
```

Contoh:

```
Username:
<input name="username" type="text">
Password:
<input name="password" type="password">
<input type="submit" value="Login">
```

Kita tidak akan membuat contoh kode program lengkap yang menunjukkan penggunaan elemen *password* karena caranya sama seperti pada elemen *text field*. Namun, hal yang perlu diingat adalah: pada saat kita melakukan *query* menggunakan metode *GET*, data yang dimasukkan melalui elemen *password* akan ditampilkan tanpa enkripsi pada URL yang dihasilkan. Maka dari itu, seharusnya elemen *password* hanya digunakan jika metode pengiriman datanya menggunakan *POST*.

Menggunakan *String*

String merupakan hal esensial yang banyak dijumpai dalam kasus-kasus pemrograman. Oleh karena itu, pengetahuan tentang *string* sangat diperlukan. Untuk kebutuhan-kebutuhan yang berkaitan dengan *string*, PHP telah menyediakan beragam fungsi yang siap digunakan.

Dalam modul ini Anda akan mempelajari materi-materi berikut:

- Menentukan format *string*
- Memperoleh panjang *string*
- Mencari atau mengambil *substring* di dalam *string*
- Menghilangkan spasi pada bagian awal dan akhir *string*
- Mengganti huruf kecil menjadi besar dan sebaliknya

Melakukan Format *String*

PHP menyediakan dua buah fungsi untuk melakukan *formatting* string, yaitu `printf()` dan `sprintf()`. Melalui fungsi-fungsi tersebut, Anda dapat menampilkan data dengan format-format tertentu, yang dianggap sebagai parameter di dalam *string*.

Contoh:

```
printf("Angka hoki Anda: %d", 4);
```

Ketika *string* di atas ditampilkan ke halaman *web*, `%d` di atas akan diganti dengan bilangan 4. `%d` merupakan format untuk menampilkan bilangan bulat (*integer*). Format-format lain yang dapat Anda gunakan dapat dilihat pada tabel di bawah ini.

Format	Keterangan
%b	Menampilkan bilangan <i>integer</i> dalam bentuk biner
%c	Menampilkan bilangan <i>integer</i> yang disamakan dengan kode ASCII
%f	Menampilkan bilangan <i>floating-point</i>
%o	Menampilkan bilangan dalam bentuk oktal
%s	Menampilkan <i>string</i>
%x	Menampilkan bilangan dalam bentuk heksadesimal menggunakan huruf kecil
%X	Menampilkan bilangan dalam bentuk heksadesimal menggunakan huruf kapital

Praktikum 10-1: Menampilkan *string* dengan format tertentu

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Melakukan Format String</title>
```

```
</head>
<body>
  <h2>Demo menampilkan string dengan format tertentu</h2>

  <?php
    printf("Bilangan bulat: %d<br />", 4);
    printf("Bilangan I: %d, Bilangan II: %d<br />", 4, 7);
    printf("Bilangan riil: %f<br />", 53.26);
    printf("String: %s<br />", "Contoh teks");
    printf("Oktal: %o<br />", 8);
    printf("Heksadesimal (huruf kecil): %x<br />", 16);
    printf("Heksadesimal (huruf besar): %X", 16);
  ?>

</body>
</html>
```

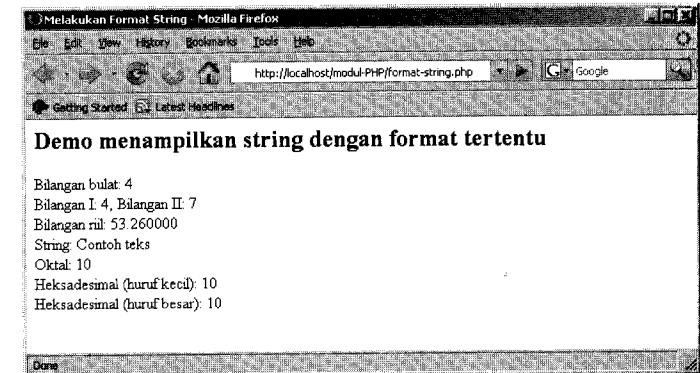
- M Simpan *file* dengan nama **format-string.php**
- M Jalankan *web browser* dan masukkan:
`http://localhost/modul-PHP/format-string.php`

Keterangan:

Seperti yang Anda lihat pada kode di atas, pada baris kedua kita membuat dua parameter yang akan ditampilkan. Dalam contoh ini, keduanya menggunakan format `%d`:

```
printf("Bilangan I: %d, Bilangan II: %d<br />", 4, 7);
```

Pada kasus seperti ini, Anda juga harus menyertakan dua buah nilai yang akan ditampilkan (dalam ini nilai 4 dan 7), yang harus dipisahkan menggunakan tanda koma.



Memperoleh Panjang String

Panjang suatu *string* adalah jumlah karakter yang menyusun *string* tersebut. Dalam PHP, kita dapat memperoleh panjang *string* menggunakan fungsi `strlen()`.

Praktikum 10-2: Memperoleh panjang string

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<html>
<head>
  <title>Memperoleh Panjang String</title>
</head>
<body>
<h2>Demo menampilkan panjang string</h2>

<?php
  $str1 = "Modul Praktikum Pemrograman Web";
  $str2 = "PHP";
  echo "\$str1: $str1<br />";
  printf("Panjang string: %d ", strlen($str1));
  echo "<br /><br />";
  echo "\$str2: $str2<br />";
  printf("Panjang string: %d ", strlen($str2));
  ?>

</body>
</html>
```

- ✓ Simpan file dengan nama **panjang-string.php**
- ✓ Jalankan *web browser* dan masukkan:
`http://localhost/modul-PHP/panjang-string.php`

Contoh:

String yang disimpan dalam variabel `$str1` memiliki panjang 31 karena spasi juga ikut dihitung sebagai satu karakter. Seperti yang Anda lihat,

kita dapat mengambil jumlah karakter dari variabel `$str1` melalui kode berikut:

```
strlen($str1)
```

Anda juga dapat melewati string secara langsung pada fungsi `strlen()`, tanpa harus menyimpannya ke dalam variabel terlebih dahulu.

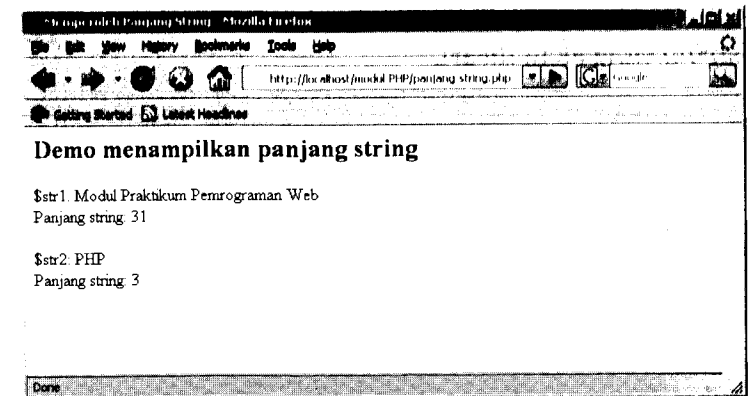
Contoh:

```
strlen("Hello World!")
strlen("PT. Multidimension Group")
```

Mencari dan Mengambil Substring dari Suatu String

Pada bagian ini Anda akan diajak mempraktikkan cara melakukan pencarian *substring* (bagian *string*) di dalam suatu *string*. Selain pencarian, Anda juga akan mempelajari cara melakukan pengambilan *substring* berdasarkan indeks tertentu.

Pencarian *substring* dilakukan menggunakan fungsi `strstr()`, sedangkan pengambilan *substring* dari suatu *string* dilakukan menggunakan fungsi `substr()`.



Fungsi `strstr()` memiliki dua buah parameter: *string* sumber dan bagian *string* yang akan dicari.

Contoh:

```
$s = strstr("Pemrograman", "gram");
```

Jika bagian *string* yang dicari tidak terdapat di dalam *string* sumber, `strstr()` akan mengembalikan nilai `false`. Sebaliknya, jika ditemukan, maka fungsi `strstr()` akan mengembalikan *string* sisa dimulai dari posisi *substring* yang ditemukan. Pada contoh di atas, fungsi `strstr()` akan mengembalikan *string* "graman".

Fungsi `substr()` memiliki tiga buah parameter: *string* sumber, indeks awal *substring*, dan jumlah karakter yang akan diambil.

Contoh:

```
$substring = substr("Pemrograman", 2, 7);
```

Pada contoh di atas, variabel `$substring` akan berisi *string* "mrograman". Jika parameter terakhir tidak disertakan, maka yang akan dikembalikan oleh fungsi `substr()` adalah "mrograman".

Praktikum 10-3: Mencari dan mengambil *substring* dalam *string*

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>strstr() dan substr()</title>
</head>
<body>
<h2>Demo mencari dan mengambil substring</h2>

<?php
```

```
$str = "Pemrograman";
echo "\$str: $str<br /><br />";

echo "<strong>Mencari Substring:</strong><br />";
$substr = strstr($str, "gram");
if ($substr) {
    echo "Subtring \"gram\" ditemukan dalam \$str ".
        "<br /><br />";
} else {
    echo "Subtring \"gram\" tidak ditemukan ".
        "dalam \$str<br /><br />";
}

echo "<strong>Mengambil Substring:</strong><br />";
echo substr($str, 2) . "<br />";
echo substr($str, 2, 7);
}

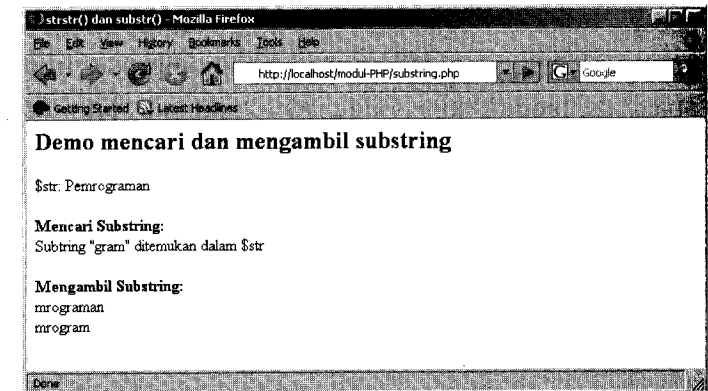
</body>
</html>
```

- 1. Simpan file dengan nama **substring.php**
- 2. Jalankan web browser dan masukkan:
<http://localhost/modul-PHP/substring.php>

Keterangan:

Pada praktikum ini mula-mula kita mencari *substring* "gram" dalam *string* "Pemrograman". Karena ditemukan, maka variabel `$str` akan bernilai "graman" sehingga program akan mengeksekusi statemen berikut:

```
echo "Subtring \"gram\" ditemukan dalam \$str ".
    "<br /><br />";
```



Selanjutnya, kita melakukan pengambilan *substring* dari *string* "Pemrograman". Pertama, kita menggunakan kode:

```
substr($str, 2)
```

Kode di atas memerintahkan program untuk mengambil *substring* mulai dari indeks ke-2 sampai akhir *string*.

Kedua, kita menggunakan kode:

```
substr($str, 2, 7)
```

Kode tersebut memerintahkan program untuk mengambil *substring* mulai dari indeks ke-2 sebanyak 7 karakter. Ingat! **Indeks string dalam PHP dimulai dari 0, bukan 1.**

Menghilangkan Spasi pada Bagian Awal dan Akhir String

Pada saat *user* melakukan *input* melalui *form* yang disediakan, ada kalanya secara tidak sengaja *user* mengetikkan spasi di bagian awal atau akhir *string*. Untuk memastikan *string* yang diproses tidak mengandung karakter spasi di bagian awal atau akhir *string*, kita dapat menghilangkannya menggunakan fungsi `ltrim()`, `rtrim()`, atau `trim()`. Sebenarnya yang dapat dihilangkan atau dihapus oleh fungsi-fungsi tersebut bukan hanya karakter spasi, tapi *white space character*, yang meliputi: karakter tab (`'\t'`) dan karakter baris baru (`'\n'`)

Fungsi `ltrim()` hanya dapat digunakan untuk menghilangkan karakter spasi yang ada di bagian awal saja. Sebaliknya, `rtrim()` hanya menghapus karakter spasi yang ada di bagian akhir saja. Untuk menghilangkan

karakter di bagian awal dan akhir, Anda perlu menggunakan fungsi `trim()`.

Contoh:

```
$str1 = "  Modul Praktikum Pemrograman Web";  
echo ltrim($str1) . "<br />";  
$str2 = "Modul Praktikum Pemrograman Web  ";  
echo rtrim($str2) . "<br />";  
$str3 = "  Modul Praktikum Pemrograman Web  ";  
echo trim($str3);
```

Praktikum 10-4: Menghilangkan spasi di bagian awal dan akhir string

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>  
<head>  
    <title>ltrim() dan trim()</title>  
</head>  
<body>  
<h2>Demo menghilangkan spasi di bagian awal  
    dan akhir string</h2>  
  
<?php  
    echo "<strong>Menggunakan ltrim():</strong><br />";  
  
    $str1 = "  Modul Praktikum Pemrograman Web";  
    echo "<strong>\$str1: \$str1<br />";  
    echo "Panjang karakter: " . strlen($str1);  
    echo "<br />";  
  
    $s1 = ltrim($str1);  
    echo "<strong>\$s1: \$s1<br />";  
    echo "Panjang karakter: " . strlen($s1);  
    echo "<br /><br />";  
  
    echo "<strong>Menggunakan trim():</strong><br />";  
  
    $str2 = "  Modul Praktikum Pemrograman Web  ";
```

```

echo "\$str2: $str2<br />";
echo "Panjang karakter: " . strlen($str2);
echo "<br />";

$s2 = trim($str2);
echo "\$s2: $s2<br />";
echo "Panjang karakter: " . strlen($s2);
?>

</body>
</html>

```

- ☑ Simpan file dengan nama **trim.php**
- ☑ Jalankan web browser dan masukkan:
http://localhost/modul-PHP/trim.php

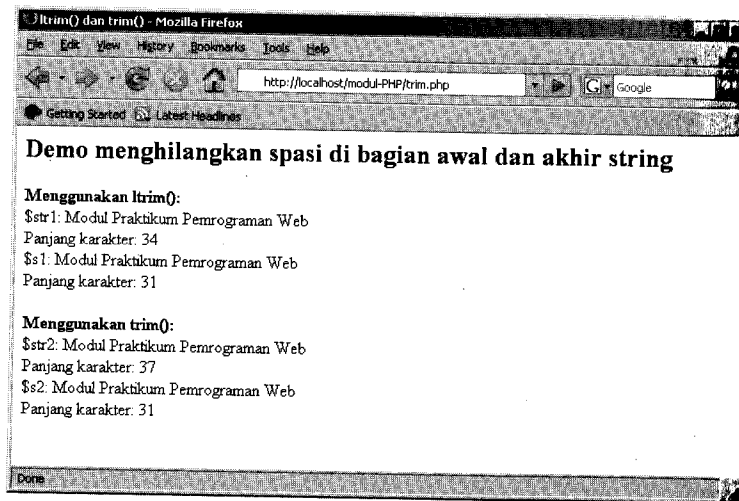
Keterangan:

Pada praktikum kali ini, kita mengisikan tiga karakter spasi pada bagian awal variabel `$str1`. Panjang string variabel `$str1` adalah 34 karakter.

Kita dapat menghilangkan karakter spasi tersebut menggunakan fungsi `ltrim()`, lalu menampung hasilnya ke dalam variabel `$s1`.

```
$s1 = ltrim($str1);
```

Melalui kode di atas, `$s1` akan berisi string tanpa karakter spasi di bagian awalnya. Dengan demikian, panjang string variabel `$s1` adalah 31 karakter.



Selanjutnya, kita mengisikan tiga karakter spasi pada bagian awal dan akhir variabel `$str2` sehingga jumlah karakter yang terkandung di dalamnya adalah 37. Karena karakter spasi yang akan dihilangkan ada di bagian awal dan akhir string, maka kita perlu menggunakan fungsi `trim()`. String yang telah dihilangkan spasinya tersebut akan ditampung ke variabel `$s2`.

```
$s2 = trim($str2);
```

Kode di atas akan menyebabkan isi dari variabel `$s2` sama dengan `$s1`.

Mengubah Huruf Kecil Menjadi Besar dan Sebaliknya

Dalam kasus-kasus pemrograman tertentu ada kalanya kita dituntut untuk mengubah jenis huruf dari huruf kecil menjadi huruf besar (kapital) maupun sebaliknya. Untuk keperluan semacam ini, PHP menyediakan fungsi `strtoupper()` dan `strtolower()`.

Fungsi `strtoupper()` berguna untuk menjadikan semua karakter di dalam suatu string menjadi huruf kapital. Fungsi `strtolower()` merupakan kebalikan dari fungsi `strtoupper()`, yang berguna untuk menjadikan semua karakter di dalam suatu string menjadi huruf kecil.

Praktikum 10-5: Mengubah huruf dalam string

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```

<html>
<head>
    <title>strtoupper() dan strtolower()</title>
</head>

```

```
<body>
<h2>Demo mengubah huruf dalam string</h2>

<?php

$nama = "BImO hArYo HaNGgoRO";

echo "string: ". $nama . "<br />";
echo "strtoupper(): " . strtoupper($nama) . "<br />";
echo "strtolower(): " . strtolower($nama) . "<br />";
?>

</body>
</html>
```

Modul 11

Menggunakan *File*

- ☑ Simpan *file* dengan nama **ubah-huruf.php**
- ☑ Jalankan *web browser* dan masukkan:
http://localhost/modul-PHP/ubah-huruf.php

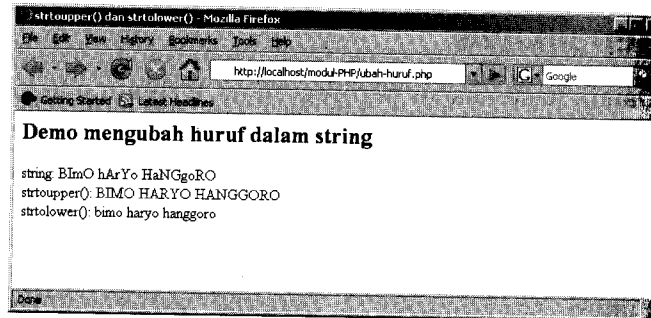
Keterangan:

Pada praktikum ini kita memasukkan string "BImO hArYo HaNGgoRO" ke dalam variabel `$nama`. Pada baris berikutnya, kita menjadikan string tersebut menjadi huruf kapital dengan memanggil fungsi `strtoupper()`.

```
echo "strtoupper(): " . strtoupper($nama) . "<br />";
```

Selanjutnya, kita juga menjadikan isi variabel `$nama` menjadi huruf kecil menggunakan fungsi `strtolower()`.

```
echo "strtolower(): " . strtolower($nama) . "<br />";
```



File dapat dijadikan alternatif untuk menyimpan data-data berukuran kecil yang dimasukkan *user* melalui *form*. *File* juga banyak digunakan untuk menyimpan data-data untuk keperluan konfigurasi, parameter, maupun yang lainnya. PHP menyediakan fasilitas untuk memudahkan Anda untuk membaca dan menulis data dari/ke dalam suatu *file*. Dalam modul ini Anda akan mempelajari materi-materi berikut:

- Membuka dan menutup *file*
- Menulis data ke dalam *file*
- Membaca isi *file* dan menampilkannya ke dalam halaman *web*
- Menggunakan *file* untuk membuat buku tamu

Membuka dan Menutup File

Sebelum Anda mengakses suatu *file* melalui aplikasi yang Anda buat untuk kemudian dibaca atau diisi data, Anda harus membuka *file* tersebut menggunakan fungsi `fopen()`. Bentuk umum penggunaan fungsi `fopen()` adalah:

```
file_handler = fopen>NamaFile, Mode
```

file_handler adalah *pointer* ke *file* yang mengace ke *file* yang Anda buka.

NamaFile adalah nama *file* yang akan diakses (lengkap dengan path-nya jika *file* tidak berada dalam satu direktori dengan *file* kode program).

Mode adalah jenis akses yang akan dilakukan, bisa berupa salah satu dari:

- **r** – Membuka *file* untuk proses pembacaan (*reading*) dengan status *read only*. Anda tidak dapat menulis apa pun ke dalam *file*.
- **r+** – Membuka *file* untuk proses pembacaan (*reading*) dan penulisan. Jika *file* belum ada maka *file* akan dibuat. Sebaliknya, jika *file* telah ada, maka data baru akan ditambahkan pada bagian awal, yaitu dengan melakukan penimpaan (*overwriting*) terhadap data lama.
- **w** – Membuka *file* untuk proses penulisan (*writting*). Jika *file* belum ada maka *file* akan dibuat. Sebaliknya, jika *file* yang didefinisikan sudah ada, maka *file* tersebut akan dihapus, selanjutnya akan dibuat *file* baru (dengan data kosong).
- **w+** – Membuka *file* untuk proses penulisan dan pembacaan. Konsepnya sama seperti pada mode **w**.
- **a** – Membuka *file* untuk proses penambahan data (*appending*). Jika *file* belum ada maka *file* akan dibuat. Sebaliknya, jika *file* telah ada maka data yang dimasukkan ke dalam *file* akan ditambahkan pada bagian akhir dari data lama (*end of file*).

- **a+** – Membuka *file* untuk proses pembacaan dan penambahan data. Konsepnya sama seperti pada mode **a**.

Jika *file* yang dibuka merupakan *file* biner (bukan *file* teks) maka Anda perlu menambahkan huruf “**b**” pada mode yang terdapat pada fungsi `fopen()`.

Contoh:

```
$file_handle = fopen($binaryfile, "rb+");
```

Ketika Anda telah selesai membaca atau menulis *file* melalui aplikasi yang Anda buat, Anda perlu menutup *file* tersebut menggunakan fungsi `fclose()`. Bentuk umum penggunaan fungsi ini adalah:

```
fclose(file_pointer)
```

Contoh:

```
fclose($file_handle);
```

Membaca Isi File

PHP menyediakan sejumlah fungsi untuk melakukan pembacaan data dari *file*. Metode yang digunakan pun beragam; ada yang dibaca per baris, per *byte*, dan ada juga yang dibaca per karakter.

Pada bagian ini Anda akan mempraktikkan cara membaca data dari suatu *file* teks dan menampilkannya ke dalam halaman *web*. Namun, sebelum Anda melakukan praktikum-praktikum di bawah ini, silakan Anda buat terlebih dahulu suatu *file* teks, misalnya dengan isi seperti berikut:

Asfira Fauziyah

Rizki Dwi Mulyana

Sebagai latihan, simpanlah file tersebut di direktori **C:\Apache2.2\htdocs\modul-PHP** dengan nama **contoh.txt**.

Praktikum 11-1: Membaca file per baris menggunakan fungsi `fgets()`

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<html>
<head>
  <title>Membaca File</title>
</head>
<body>
<h2>Demo membaca file (Bagian 1)</h2>

<?php
$file = "c:\\Apache2.2\\htdocs\\modul-PHP\\contoh.txt";
$fp = fopen($file, "r");

echo "Isi file $file adalah:<br />";
while (!feof($fp)) {
    $baris = fgets($fp, 1024);
    echo "<strong>$baris</strong><br />";
}
fclose($fp);
?>

</body>
</html>
```

- ✓ Simpan file dengan nama **bacafile1.php**
- ✓ Jalankan web browser dan masukkan **http://localhost/modul-PHP/bacafile1.php**

Keterangan:

Pada praktikum di atas kita membaca file dengan cara mengambilnya baris demi baris, yaitu dengan menggunakan fungsi `fgets()`. Setiap baris yang dibaca akan

ditampung ke dalam variabel `$baris` dan kemudian ditampilkan ke halaman web menggunakan perintah `echo`. Proses ini akan diulang (menggunakan statemen `while`) sampai fungsi `feof()` mengembalikan nilai `true`. Fungsi `feof()` di atas berguna untuk memeriksa apakah proses pembacaan sudah sampai ke bagian akhir file atau belum. Nilai 1024 yang terdapat pada pemanggilan fungsi `fgets()` adalah nilai byte maksimal yang dapat dibaca pada satu baris data tertentu. Anda dapat mengubah nilai tersebut sesuai dengan kebutuhan.

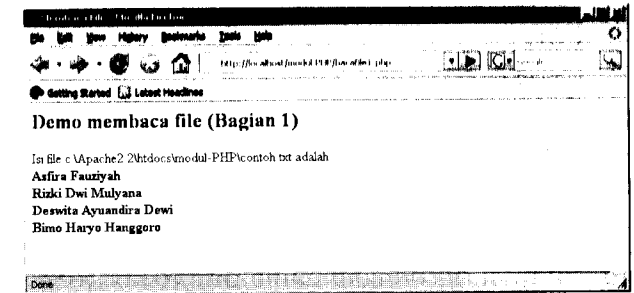
Praktikum 11-2: Membaca file per n byte menggunakan fungsi `fread()`

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

```
<html>
<head>
  <title>Membaca File</title>
</head>
<body>
<h2>Demo membaca file (Bagian 2)</h2>

<?php
$file = "c:\\Apache2.2\\htdocs\\modul-PHP\\contoh.txt";
$fp = fopen($file, "r");

echo "Isi file $file adalah:<br />";
while (!feof($fp)) {
    $pecahandata = fread($fp, 30);
```



```

echo "<strong>$specahandata</strong><br />";
}
fclose($fp);
?>

</body>
</html>

```

- ☒ Simpan file dengan nama **bacafile2.php**
- ☒ Jalankan web browser dan masukkan **http://localhost/modul-PHP/bacafile2.php**

Keterangan:

Kali ini, kita menggunakan fungsi `fread()` dalam membaca isi file. Fungsi `fread()` akan membaca

sejumlah byte dari isi file. Pada contoh di atas, kita memasukkan nilai 30 sebagai ukuran byte yang akan kita baca.

```
$specahandata = fread($fp, 30);
```

Dengan demikian, pada setiap pengulangan yang dilakukan di atas, `fread()` akan mengambil 30 byte data dari file **contoh.txt** dan menampilkannya ke layar web browser.

Karena tampilan yang dihasilkan tidak sesuai dengan yang kita harapkan, kita dapat mengubah kode di atas menjadi seperti berikut:

```

<html>
<head>
    <title>Membaca File</title>
</head>

```

```

<body>
<h2>Demo membaca file (Bagian 2)</h2>

<?php
$file = "c:\\Apache2.2\\htdocs\\modul-PHP\\contoh.txt";
$fp = fopen($file, "r");

echo "Isi file $file adalah:<br />";
$specahandata = fread($fp, filesize($file));
echo "<pre><strong>$specahandata</strong></pre>";
fclose($fp);
?>

</body>
</html>

```

Pada modifikasi kode di atas, kita membaca byte data sebanyak ukuran file. Untuk mengetahui ukuran dari suatu file, gunakan fungsi `filesize()`. Maka dari itu, kita tidak perlu lagi melakukan pengulangan. Selanjutnya, agar data yang ditampilkan di dalam web browser sama persis seperti tampilan yang terdapat di dalam file, pada contoh ini kita menggunakan tag `<pre>`.

Praktikum 11-3: Membaca file per karakter menggunakan fungsi `fgetc()`

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```

<html>
<head>
    <title>Membaca File</title>
</head>
<body>
<h2>Demo membaca file (Bagian 3)</h2>

<?php
$file = "c:\\Apache2.2\\htdocs\\modul-PHP\\contoh.txt";
$fp = fopen($file, "r");

```

```

echo "Isi file $file adalah:<br />";
while (!feof($fp)) {
    $karakter = fgetc($fp);
    echo "<strong>$karakter</strong>";
}
fclose($fp);
?>

</body>
</html>

```

- ❑ Simpan *file* dengan nama **bacafile3.php**
- ❑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/bacafile3.php**

Keterangan:

Fungsi lain yang dapat digunakan untuk membaca data suatu file adalah `fgetc()`, yang akan membaca data per karakter. Pada contoh di atas, kita menangkap setiap karakter yang dibaca ke dalam variabel `$karakter`, menggunakan kode berikut:

```
$karakter = fgetc($fp);
```

Proses ini akan diulang sampai fungsi `feof()` menghasilkan nilai `true`.

Menulis atau Menambahkan Data ke dalam File

Proses penulisan (*writting*) dan penambahan (*appending*) data ke dalam suatu file sebenarnya merupakan proses yang sama. Perbedaannya hanya terletak pada saat pemanggilan fungsi `fopen()`. Pada saat Anda ingin

menulis ke *file*, Anda seharusnya menggunakan mode "**w**" ketika memanggil fungsi `fopen()`.

```
$fp = fopen($file, "w");
```

Pada saat Anda ingin menambahkan data pada suatu *file* yang sebelumnya sudah ada datanya, Anda tinggal menggunakan mode "**a**".

```
$fp = fopen($file, "a");
```

PHP menyediakan beberapa fungsi untuk keperluan penulisan data ke dalam *file*. Namun, di sini kita hanya akan menggunakan fungsi `fwrite()` dan `fputs()`.

Praktikum 11-4: Menulis data ke dalam file menggunakan fungsi `fwrite()` dan `fputs()`

- ❑ Jalankan aplikasi Text Editor
- ❑ Tuliskan kode berikut:

```

<html>
<head>
    <title>Menulis File</title>
</head>
<body>
    <h2>Demo menulis dan menambah data ke dalam file</h2>

<?php
$file = "c:\\Apache2.2\\htdocs\\modul-PHP\\contoh1.txt";

echo "<p>Menulis data ke $file...</p>";
$fp = fopen($file, "w");
fwrite($fp, "Data yang ditulis ke dalam file.\n");
fclose($fp);

echo "<p>Menambah data ke $file...</p>";
$fp = fopen($file, "a");
fputs($fp, "Data yang ditambah ke dalam file.\n");
fclose($fp);

```

```

echo "Isi file $file adalah:<br />";
$fp = fopen($file, "r");
while (!feof($fp)) {
    $baris = fgets($fp, 1024);
    echo "<strong>$baris</strong><br />";
}
fclose($fp);
?>

</body>
</html>

```

- ☑ Simpan file dengan nama **tulisfile.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/tulisfile.php**

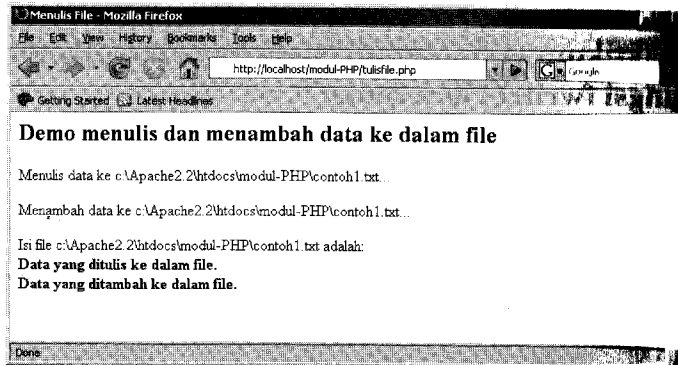
Keterangan:

Pada praktikum kali ini, mula-mula kita membuat file baru dengan nama **contoh1.txt** dan menempatkannya di direktori

c:\Apache2.2\htdocs\modul-PHP. Proses ini dibuat dengan cara melewati mode "w" pada saat pemanggilan fungsi `fopen()`. Selanjutnya, kita menulis data ke dalam file tersebut menggunakan fungsi `fwrite()`:

```
fwrite($fp, "Data yang ditulis ke dalam file.\n");
```

Ketika selesai menulis file, kita menutup koneksi dengan file tersebut menggunakan fungsi `fclose()`.



Tangkah selanjutnya, kita membuka kembali file yang telah dibuat dan ditulis di atas untuk ditambah data yang kita inginkan. Untuk keperluan ini, Anda perlu menggunakan mode "a" pada saat memanggil fungsi `fopen()`. Dalam contoh ini, kita menggunakan fungsi `fputs()` untuk menambah data ke dalam file:

```
fputs($fp, "Data yang ditambah ke dalam file.\n");
```

Setelah selesai ditambah datanya, koneksi file kembali ditutup menggunakan fungsi `fclose()`.

Untuk melihat isi file yang telah terbuat, kita dapat melakukan pembacaan terhadap file tersebut menggunakan fungsi `fgets()`.

Menggunakan File untuk Membuat Buku Tamu

Setelah Anda memahami konsep pembacaan dan penulisan file, pada bagian ini kita akan mengimplementasikan konsep tersebut untuk membuat aplikasi buku tamu sederhana.

Praktikum 11-5: Menggunakan file untuk membuat buku tamu

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```

<html>
<head>
    <title>Menggunakan File</title>
</head>
<body>
<h2>Demo menggunakan file untuk membuat buku tamu</h2>
<form action="bukutamu.php" method="post">
<table>
    <tr>
        <td valign="top">Nama</td>
        <td valign="top">:</td>

```

```

<td><input type="text" name="nama"
size="40" maxlength="25" /></td>
</tr>
<tr>
<td valign="top">Email</td>
<td valign="top">:</td>
<td><input type="text" name="email"
size="40" maxlength="40" /></td>
</tr>
<tr>
<td valign="top">Komentar</td>
<td valign="top">:</td>
<td><textarea name="komentar"
cols="35" rows="5"></textarea></td>
</tr>
<tr>
<td></td>
<td></td>
<td><input type="submit" name="kirim"
Value="Kirim Komentar" /></td>
</tr>
</table>
</form>

</body>
</html>

```

- ☑ Simpan *file* dengan nama **form-bukutamu.html**
- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```

<html>
<head>
<title>Menggunakan File</title>
</head>
<body>
<h2>Demo menggunakan file untuk membuat buku tamu</h2>

<?php
if (isset($_POST['kirim'])) {
    $nama = $_POST['nama'];
    $email = $_POST['email'];
    $komentar = $_POST['komentar'];

    $file = "c:\\Apache2.2\\htdocs",

```

```

"\\modul-PHP\\bukutamu.txt");

if (!file_exists($file)) {
    $fp = fopen($file, "w");
} else {
    $fp = fopen($file, "a");
}

$barisdata = $nama .";". $email .";". $komentar . "\n";
fputs($fp, $barisdata);

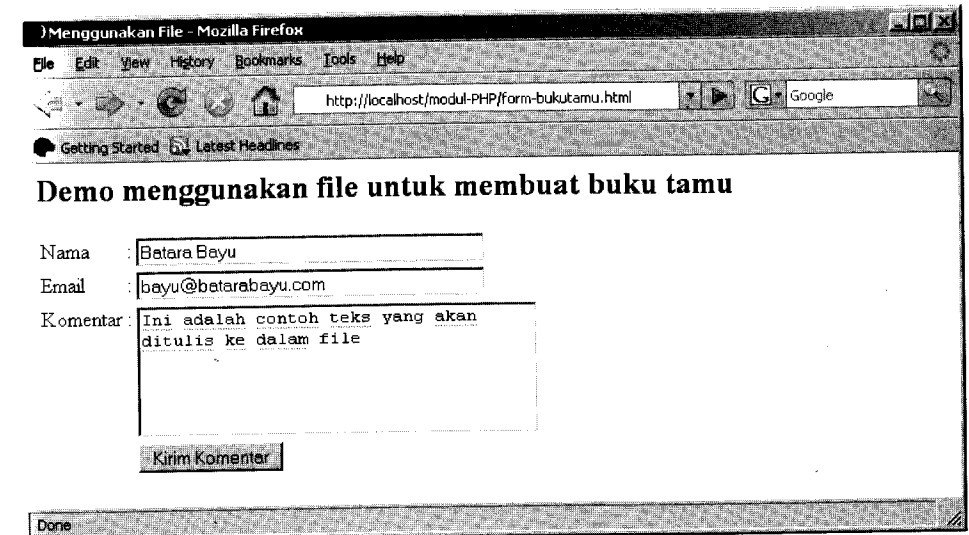
fclose($fp);

echo "Terima kasih <strong>$nama</strong>, ".
      "komentar Anda sudah tersimpan di server kami";
}
}

</body>
</html>

```

- 1. Simpan *file* dengan nama **bukutamu.php**
- 2. Jalankan *web browser* dan masukkan:
http://localhost/modul-PHP/form-bukutamu.html



Keterangan:

Pada praktikum ini, kita menangkap data-data yang dimasukkan melalui form dan memasukkannya ke dalam file. Mula-mula kita memeriksa apakah file sudah terbuat di direktori yang didefinisikan atau belum, yaitu dengan menggunakan fungsi `file_exists()`. Jika belum, kita melewati mode "w" ke dalam fungsi `fopen()`. Ini berarti file akan dibuat terlebih dahulu sebelum akhirnya ditulis. Jika file sudah ada, maka kita hanya perlu menambahkan data ke dalam file tersebut, yaitu dengan cara melewati mode "a" ke dalam fungsi `fopen()`.

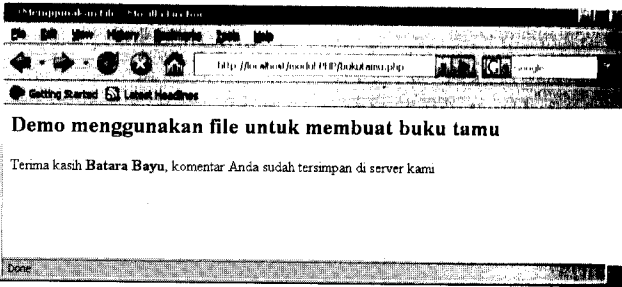
```
if (!file_exists($file)) {
    $fp = fopen($file, "w");
} else {
    $fp = fopen($file, "a");
}
```

Dalam contoh ini, data yang dimasukkan akan dijadikan sebagai satu baris di dalam file. Untuk keperluan seperti ini, kita menggunakan kode berikut:

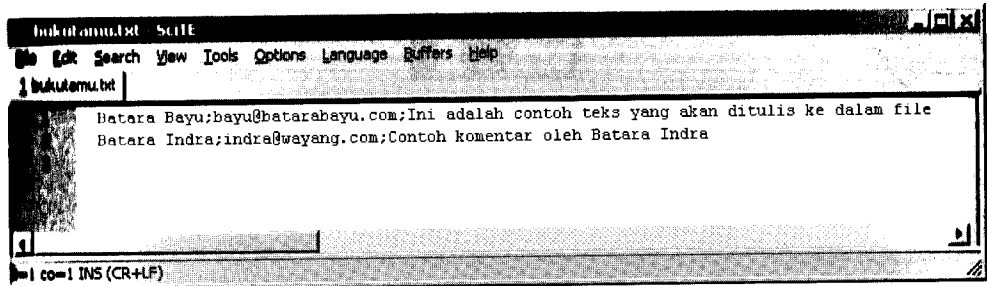
```
$barisdata = $nama .";". $email .";". $komentar . "\n";
fputs($fp, $barisdata);
```

Seperti yang Anda lihat pada kode di atas, antara data nama, email, dan komentar masing-masing akan dipisahkan oleh tanda titik koma atau semicolon (;).

Setelah selesai menulis atau menambah data ke dalam file, kita menutup file dengan menggunakan `fclose()`.



Anda dapat memeriksa isi file **bukutamu.txt** yang terdapat di direktori **C:\Apache2.2\htdocs\modul-PHP**. Contoh isinya adalah seperti berikut:



Melakukan *Upload File*

Proses *upload* adalah menyalin *file* dari direktori lokal ke komputer *server*. Sebagai contoh sederhana, pada saat Anda ingin menampilkan foto Anda di internet, maka *file* foto tersebut harus di-*upload* terlebih dahulu ke suatu *server* tertentu.

Fasilitas *upload* itu sendiri ditujukan untuk memudahkan *user* dalam melakukan penyalinan *file* dari komputer lokal ke komputer *server* melalui *user-interface* yang sangat nyaman. *User* tinggal menunjuk *file* yang akan di-*upload*, kemudian klik tombol **SUMBIT** yang disediakan, tanpa harus

usah-susah menuliskan perintah 'ini itu' di dalam *command-prompt* atau *hell*.

Dalam modul ini Anda akan mempelajari materi-materi berikut:

- Mengenal tipe *FILE* dalam tag `<input>`
- Memahami cara kerja *upload*
- Menentukan lokasi tujuan proses *upload*
- Menyiapkan *form* untuk proses *upload*
- Menulis kode PHP untuk melakukan *upload file*
- Mengakses *file* yang telah di-*upload*

Tipe *FILE* dalam Tag `<input>`

Untuk melakukan *upload file*, kita perlu menggunakan elemen kontrol khusus. Elemen kontrol ini sebenarnya hampir sama seperti elemen *TEXT FIELD*, hanya saja di sebelah kanannya terdapat tombol **BROWSE** yang digunakan untuk menentukan lokasi *file* yang akan di-*upload*. Elemen kontrol ini dibuat menggunakan tag `<input>`, dengan atribut *TYPE* diisi dengan nilai *FILE*.

Contoh:

```
<input type="file" name="file_user">
```

Perlu diperhatikan bahwa sebelum membuat kontrol di atas, Anda perlu membuat terlebih dahulu sebuah elemen *HIDDEN CONTROL* untuk melewati nilai dari *MAX_FILE_SIZE*. *MAX_FILE_SIZE* adalah suatu variabel yang berfungsi untuk menentukan ukuran *file* maksimum (dalam satuan *byte*) yang diperbolehkan dalam proses *upload*.

Contoh:

```
<input type="hidden" name="MAX_FILE_SIZE" value="500000">
```

Cara Kerja dalam Proses *Upload*

Cara kerja dari proses *upload file* ke *web server* melalui *form HTML* selama ini masih menjadi teka-teki bagi sebagian orang. Maka dari itu, sebelum Anda mulai membuat aplikasi yang dapat melakukan proses *upload*, terlebih dahulu Anda perlu memahami cara kerja atau proses yang akan terjadi di dalamnya.

Berikut ini cara kerja dari proses *upload*:

1. User mengakses halaman web yang berisi *form upload* (yaitu *form* yang mengandung elemen *TEXT FIELD* dengan tombol **BROWSE** di sebelah kanannya)
2. User akan memilih atau menentukan *file* yang akan di-*upload* dari komputer mereka
3. Nama *file* beserta *path* lengkapnya akan muncul di dalam elemen *TEXT FIELD*
4. User melakukan klik terhadap tombol *SUBMIT* (dalam hal ini, tombol *Upload*)
5. *File* akan ditransfer ke *web server*, lalu *file* tersebut akan bertempat di direktori temporari
6. Kode PHP yang diisi pada bagian *ACTION* dari *form* di atas akan memeriksa apakah *file* tersebut sudah terkirim ke *web server* atau belum. Jika sudah, kode tersebut akan mengeksekusi perintah `copy()` untuk memindahkan *file* dari direktori temporari ke direktori sebenarnya (direktori tujuan *upload*) yang berada di dalam *web server*
7. Kode PHP memberikan konfirmasi ke *user*, apakah *file* berhasil di-*upload* atau tidak

Menentukan Lokasi *Upload*

Dalam mempelajari modul ini kita menggunakan komputer lokal untuk membuat contoh-contoh program. Maka dari itu, kita perlu melakukan

konfigurasi terhadap *file* php.ini untuk menentukan lokasi mana yang akan dijadikan sebagai direktori temporer.

Untuk melakukan hal tersebut, ikuti langkah-langkah berikut:

1. Buka *file* **php.ini** menggunakan aplikasi Text Editor
2. Cari baris yang berisi kode berikut:

```
; Temporary directory for HTTP uploaded files (will use system
default if not
specified).
```

```
upload_tmp_dir = "C:\PHP-Upload"
```

3. Ubah direktori **C:\PHP-Upload** ke lokasi yang Anda inginkan (misalnya: **D:\temp** atau yang lainnya). Jika tidak diisi, *file* yang di-*upload* akan disalin ke direktori sistem.

Menyiapkan Form untuk Proses Upload

Pada saat membuat *form* untuk menangani proses *upload*, kita perlu menyertakan atribut ENCTYPE pada tag <form>, dan isi nilainya adalah: "**multipart/form-data**". Nilai tersebut menandakan bahwa yang akan dikirimkan ke *server* bukan hanya data (teks) yang tertulis pada elemen kontrol, tetapi juga isi *file*-nya.

Praktikum 12-1: Membuat form untuk proses upload

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Form upload</title>
</head>
<body>
  <h2>Demo membuat form untuk proses upload</h2>
```

```
<ol>
  <li>Tulis nama file yang akan di-upload atau gunakan
    tombol BROWSE untuk menentukan lokasi file</li>
  <li>Ketika selesai, klik tombol
    <strong>Upload File</strong></li>
</ol>
<hr />
<div align="center">
  <form action="UploadFile.php" method="post"
    enctype="multipart/form-data">
    <input type="hidden" name="MAX_FILE_SIZE"
      value="500000"><br />
    <input type="file" name="file_user" size="60">
    <br /><br />
    <input type="submit" value="Upload File">
  </form>
</div>

</body>
</html>
```

- ☑ Simpan *file* dengan nama **upload.html**
- ☑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/upload.html**

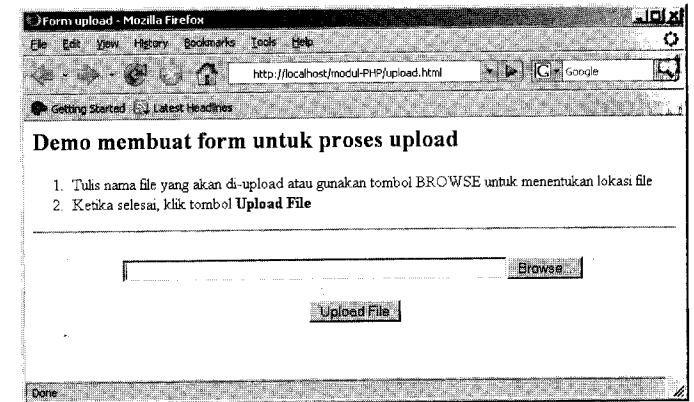
Keterangan:

Seperti yang Anda lihat pada kode di atas bahwa kita harus melewati nilai

MAX_FILE_SIZE membuat kontrol dengan tipe **FILE**.

Pada contoh tersebut,

kita menentukan nilai 500000 ke dalam **MAX_FILE_SIZE**. Ini berarti ukuran maksimum dari *file* yang akan di-*upload* adalah 500000 byte (kira-kira 488 KB). Nilai ini harus disesuaikan dengan nilai



upload_max_filesize yang terdapat pada file **php.ini**. Jika nilai **MAX_FILE_SIZE** lebih besar dari **upload_max_filesize**, maka Anda perlu mengubah secara manual nilai **upload_max_filesize** agar lebih tinggi dari nilai **MAX_FILE_SIZE**. Nilai default untuk **upload_max_filesize** adalah **2M**.

Selain **upload_max_filesize**, ada juga konfigurasi lain di dalam **php.ini** yang perlu diperhatikan kaitannya dengan proses upload, yaitu **post_max_size**. Nilai **post_max_size** adalah ukuran maksimum yang diperbolehkan dalam satu kali pengiriman. Nilai default untuk **post_max_size** adalah **8M**.

Kode PHP untuk Proses Upload

Setelah **form upload** disediakan, langkah selanjutnya adalah membuat kode yang dapat menjalankan prosesnya. Pada **form** di atas, kita menunjuk ke file **UploadFile.php** untuk melaksanakan proses **upload**. Maka dari itu, pada bagian ini kita akan membuat **file** tersebut.

Praktikum 12-2: Membuat kode PHP untuk menjalankan proses upload

- ✓ Jalankan aplikasi Text Editor
- ✓ Tuliskan kode berikut:

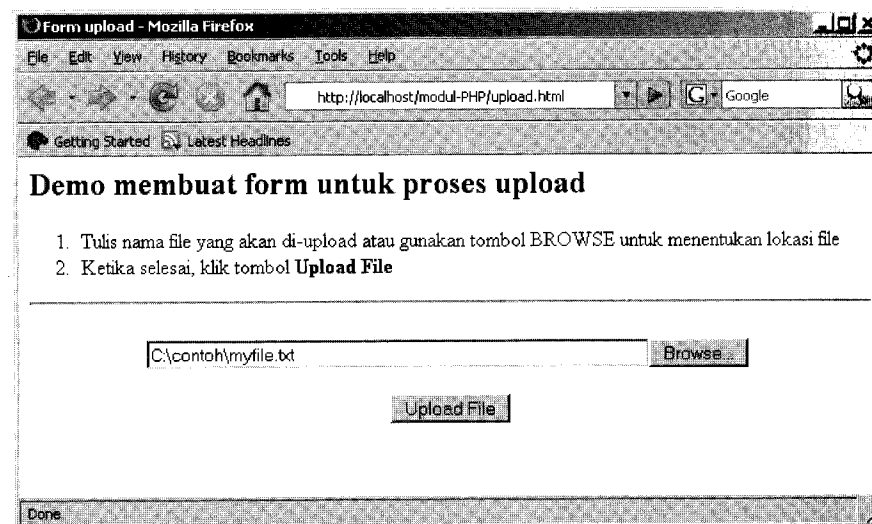
```
<?php
$file_user = $_FILES['file_user']['tmp_name'];
$namefile = $_FILES['file_user']['name'];

$tujuan = "c:\\data\\" . $namefile;

if ($file_user != "none") {
    copy($file_user, $tujuan);
    echo "<h3>Proses upload telah dilakukan.</h3>";
    echo "\$file_user: $file_user<br />";
}
```

```
echo "\$tujuan: $tujuan";
} else {
    echo "<h3>Upload Error!</h3>";
    echo "Anda belum memilih file yang akan di-upload."
}
?>
```

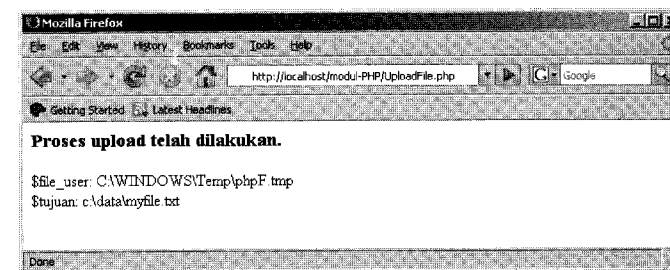
- ✓ Simpan **file** dengan nama **UploadFile.php**
- ✓ Jalankan **web browser** dan masukkan **http://localhost/modul-PHP/upload.html**
- ✓ Pilih salah satu **file** dari komputer Anda, lalu klik tombol **Upload File**



Keterangan:

File yang telah terkirim ke web server dapat diakses informasinya melalui kode berikut:

```
$file_user = $_FILES['file_user']['tmp_name'];
$namefile = $_FILES['file_user']['name'];
```



`$FILES['file_user']['tmp_name']` digunakan untuk mengambil nama file temporari beserta direktorinya, sedangkan `$FILES['file_user']['name']` untuk mengambil nama asli dari file yang diupload. Nama file temporari akan dinamai secara acak oleh sistem dan akan segera dihapus setelah PHP selesai mengeksekusi kode yang ada pada file **UploadFile.php**.

Dalam melakukan proses upload, selain `$FILES['file_user']['tmp_name']` dan `$FILES['file_user']['tmp_name']`, kita juga dapat menggunakan:

- `$FILES['file_user']['size']`, untuk mengambil ukuran file yang di-upload
- `$FILES['file_user']['tmp_name']`, untuk mengambil tipe MIME dari file yang di-upload, misalnya: image/jpeg, image/gif, dan sebagainya.

Pada baris selanjutnya, variabel `$tujuan` digunakan sebagai lokasi file tujuan upload yang sesungguhnya.

```
$tujuan = "c:\\data\\" . $namafile;
```

Maka dari itu, diperlukan proses penyalinan dari file temporari ke file tujuan. Kode yang digunakan untuk keperluan ini adalah:

```
copy($file_user, $tujuan);
```

Tujuan akhir dari praktikum kita kali ini adalah melakukan penyalinan file dari **c:\contoh\myfile.txt** (di komputer lokal) ke **c:\data\myfile.txt** (di web server/komputer server).

Melakukan Upload Beberapa File Sekaligus

Setelah Anda memahami konsep dan kode untuk melakukan proses *upload*, di sini Anda akan diajak untuk mempraktikkan cara mengembangkan contoh aplikasi *web* yang mampu melakukan *upload* lebih dari satu file secara bersamaan (satu kali tombol SUBMIT).

Praktikum 12-3: Melakukan upload beberapa file sekaligus

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Form upload</title>
</head>
<body>
  <h2>Demo melakukan upload terhadap banyak file</h2>

  <ol>
    <li>Tulis nama file yang akan di-upload atau gunakan
      tombol BROWSE untuk menentukan lokasi file
      (maksimal 5 file)</li>
    <li>Ketika selesai, klik tombol
      <strong>Upload File</strong></li>
  </ol>
  <hr />

  <div align="center">
    <form action="multiple-UploadFile.php" method="post"
      enctype="multipart/form-data">
      <?php
        for ($i=0; $i<5; $i++) {
          ?>
            <input type="hidden" name="MAX_FILE_SIZE"
              value="1000000"><br />
            <input type="file" name="file_user<?php echo $i; ?>"
              size="60">
          <?php
        }
      ?>
```

```

<br /><br />
<input type="submit" value="Upload File">
</form>
</div>

</body>
</html>

```

- ☑ Simpan *file* dengan nama **multiple-upload.html**
- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```

<?php
echo "<h3>Daftar file yang telah ter-upload:</h3>";
$counter = 0;
for ($i=0; $i<5; $i++) {
    $str = 'file_user' . $i;
    $file_user = $_FILES[$str]['tmp_name'];
    $namafile = $_FILES[$str]['name'];
    $ukuran = $_FILES[$str]['size'];
    $tipe_mime = $_FILES[$str]['type'];

    $tujuan = "c:\\data\\" . $namafile;

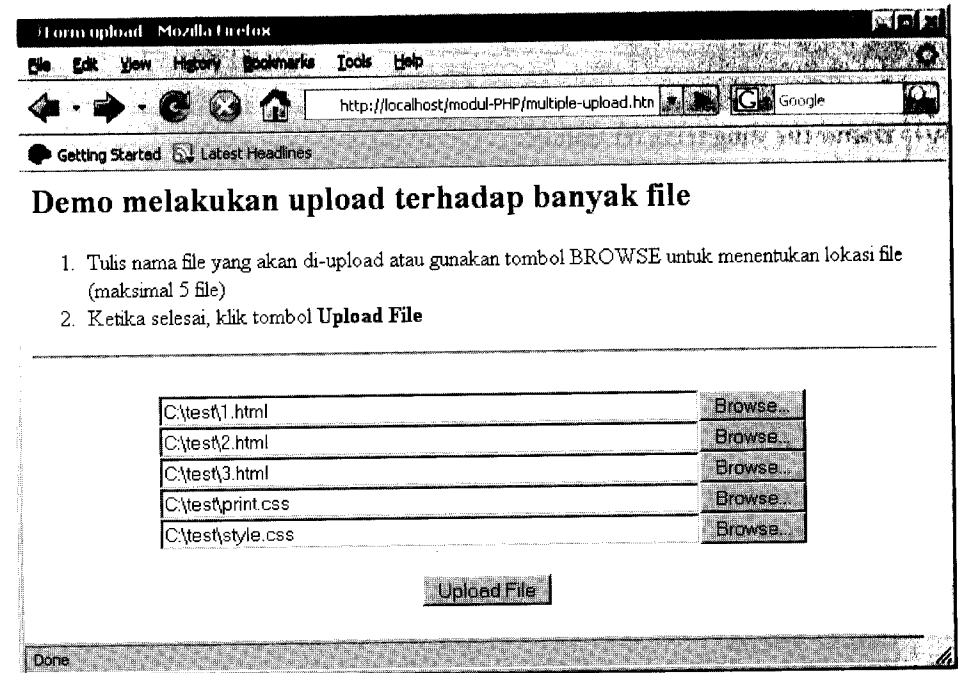
    if ($file_user != "none") {
        copy($file_user, $tujuan);

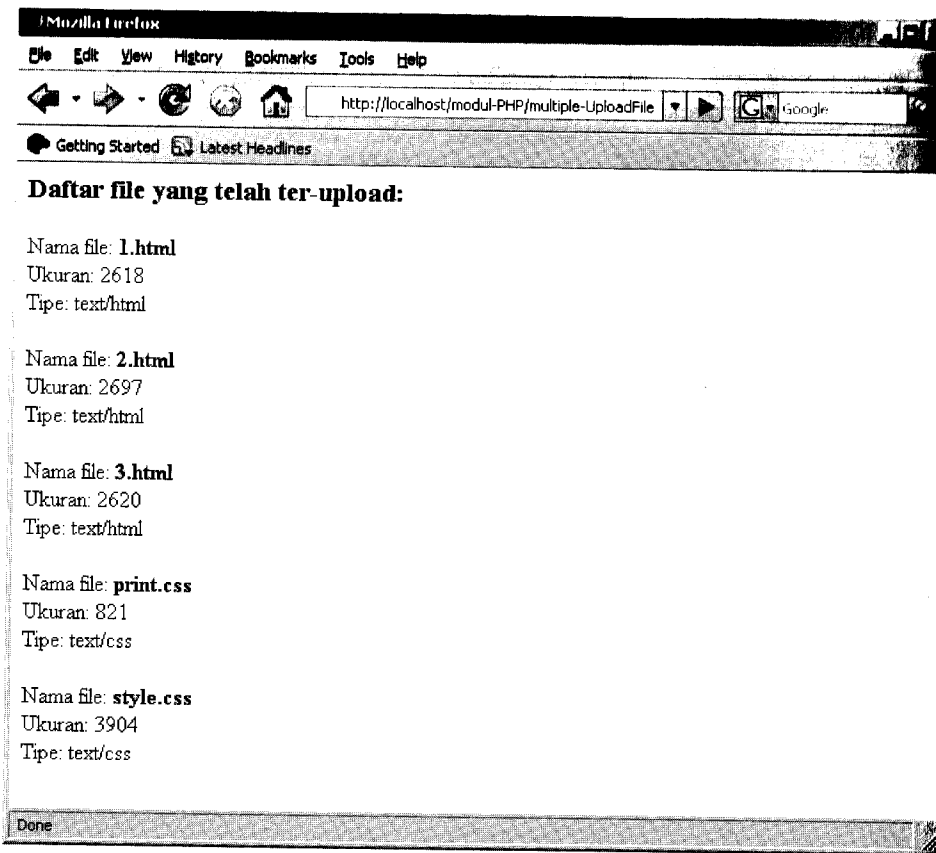
        echo "Nama file: <strong>$namafile</strong><br />";
        echo "Ukuran: $ukuran<br />";
        echo "Tipe: $tipe_mime<br /><br />";
        $counter++;
    } else {
        continue;
    }
}
if ($counter == 0) {
    echo "Tidak ada file yang ter-upload.";
}
?>

```

- ☑ Simpan *file* dengan nama **multiple-UploadFile.php**
- ☑ Jalankan *web browser* dan masukkan:
<http://localhost/modul-PHP/multiple-upload.html>

- ☑ Pilih 5 *file* dari komputer Anda, lalu klik tombol **Upload File**





Keterangan:

Pada praktikum ini sebenarnya tidak banyak kode yang diubah. Di sini, kita hanya menambahkan proses pengulangan pada saat pembuatan form maupun pada saat pemrosesan file-nya.

Pada file `multiple-upload.html`:

```
<?php
for ($i=0; $i<5; $i++) {
?>


```

```
size="60">
?php
}
```

Kode di atas berfungsi untuk membuat 5 buah elemen kontrol dengan tipe `FILE` sehingga user dapat dengan mudah mengirimkan 5 file sekaligus ke server. Dalam contoh ini, kita tidak menggunakan array sebagai nama elemen kontrol. Nama setiap elemen kontrol yang ada dalam form kita buat unik, yaitu: `file_user0`, `file_user1`, ..., `file_user4`. Kode untuk penamaannya sederhana:

```
name="file_user<?php echo $i; ?>"
```

Dalam file **`multiple-UploadFile.php`** kita menangkap nama dari setiap kontrol tersebut melalui kode:

```
$str = 'file_user' . $i;
```

Variabel `$counter` yang kita definisikan dalam file `multiple-UploadFile.php` berfungsi untuk mengetahui jumlah file yang berhasil di-upload ke server.

Dalam contoh ini, selain nama file, kita juga menampilkan ukuran dan tipe dari file tersebut dengan menggunakan kode berikut:

```
$namafile = $_FILES[str]['name'];
$ukuran = $_FILES[str]['size'];
$tipe_mime = $_FILES[str]['type'];
```

Modul
13

Mengakses *Database MySQL*

Hampir semua aplikasi *web* yang dikembangkan saat ini membutuhkan teknologi *database* untuk menyimpan dan mengelola data-data yang digunakan di dalamnya. PHP memberikan dukungan terhadap banyak jenis *database*, baik yang bersifat komersial maupun yang tidak. Dalam modul ini kita memilih MySQL sebagai *server database* yang akan kita gunakan untuk mengembangkan aplikasi.

Dalam modul ini Anda akan mempelajari materi-materi berikut:

- Mengetahui sekilas tentang MySQL
- Membangun koneksi antara PHP dan MySQL

- Memanipulasi data dalam MySQL melalui PHP
- Menampilkan data dari MySQL ke halaman *web*

Mengenal Sekilas tentang MySQL

MySQL merupakan sistem *database* yang banyak digunakan untuk pengembangan aplikasi *web*. Alasannya mungkin karena gratis, pengelolaan datanya sederhana, memiliki tingkat keamanan yang bagus, mudah diperoleh, dan lain-lain. Sebelum kita menginjak ke pembahasan lebih jauh tentang koneksi MySQL dan PHP, sekarang Anda akan diajak untuk mengenal terlebih dahulu tentang tata cara penggunaan MySQL itu sendiri.

Menjalankan MySQL Client

Untuk menjalankan MySQL *Client*, tuliskan perintah berikut pada *command-prompt*:

```
> mysql -uUSER -pPASSWORD -hHOST
```

Ganti argumen USER, PASSWORD, dan HOST dengan argumen yang sesuai dengan *setting* yang Anda lakukan untuk *database* yang Anda buat. Sebagai contoh:

```
> mysql -upras -ps3cr3T -hlocalhost
```

Semua argumen di atas sebenarnya bersifat opsional. Jika tidak disertakan, maka akan dianggap:

```
-u    shell account username
-p    no password
-h    localhost
```

Jika MySQL *Client* sudah berhasil dijalankan, pesan yang akan Anda terima adalah:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.24a-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Membuat dan Memilih Database yang Akan Digunakan

Untuk membuat *database*, gunakan perintah CREATE DATABASE dan diikuti nama *database* yang akan dibuat.

```
mysql> CREATE DATABASE CONTOHDB;
```

```
Query OK, 1 row affected (0.02 sec)
```

Untuk menampilkan daftar *database* yang terdapat di dalam *server* MySQL, gunakan perintah SHOW DATABASES:

```
mysql> SHOW DATABASES;
```

```
+-----+
| Database |
+-----+
| information_schema |
| contohdb |
| mysql |
| test |
+-----+
5 rows in set (0.00 sec)
```

Untuk memilih salah satu *database* yang diinginkan, gunakan perintah `USE` dan diikuti nama *database*-nya.

```
mysql> USE contohdb;  
Database changed
```

Menampilkan Informasi Metadata

Untuk menampilkan semua tabel yang terdapat di dalam *database*, gunakan `SHOW TABLES`.

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -proot  
mysql> show tables;  
+-----+  
Tables_in_mysql  
+-----+  
columns_priv  
db  
func  
help_category  
help_keyword  
help_relation  
help_topic  
host  
proc  
procs_priv  
tables_priv  
time_zone  
time_zone_leap_second  
time_zone_name  
time_zone_transition  
time_zone_transition_type  
user  
+-----+  
17 rows in set (0.00 sec)  
  
mysql>
```

Untuk menampilkan daftar kolom (*field*) dari suatu tabel, gunakan perintah `DESC` yang diikuti nama tabelnya.

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -proot  
mysql> desc user;  
+-----+-----+-----+-----+  
Field | Type | Collation | Index |  
+-----+-----+-----+-----+  
id     | int(4) |            | PRIMARY |  
name   | char(60) |            |          |  
password | char(16) |            |          |  
disabled | char(41) |            |          |  
select_priv | enum('N','Y') |            |          |  
insert_priv | enum('N','Y') |            |          |  
update_priv | enum('N','Y') |            |          |  
delete_priv | enum('N','Y') |            |          |  
create_priv | enum('N','Y') |            |          |  
drop_priv  | enum('N','Y') |            |          |  
+-----+-----+-----+-----+
```

Untuk menampilkan data dari suatu tabel, gunakan perintah `SELECT`, seperti yang ditunjukkan oleh contoh kode berikut:

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -proot  
mysql> select User from user where host='localhost';  
+-----+  
User |  
+-----+  
root |  
+-----+  
1 row in set (0.02 sec)  
  
mysql>
```

Memanipulasi Data

Untuk memanipulasi data pada tabel-tabel yang terdapat di dalam suatu *database*, Anda perlu mempelajari perintah-perintah SQL dari MySQL (lihat **Lampiran B – Referensi MySQL**). Berikut ini perintah-perintah yang perlu Anda pelajari secara lebih detail.

- `SELECT`: digunakan untuk mengambil data dari *database*.
- `DELETE`: digunakan untuk menghapus data dari *database*.
- `INSERT`: digunakan untuk memasukkan data baru ke dalam *database*.

- REPLACE: digunakan untuk mengganti data di dalam *database*. Jika terdapat *record* yang sama dalam suatu tabel, perintah ini akan menimpa *record* tersebut dengan data baru.
- UPDATE: digunakan untuk mengubah data di dalam suatu tabel.

Perintah-perintah di atas hanya digunakan untuk memanipulasi data. Untuk memanipulasi struktur objek *database*, gunakan perintah-perintah berikut:

- CREATE: digunakan untuk membuat *database*, tabel, atau indeks.
- ALTER: digunakan untuk memodifikasi struktur dari suatu tabel.
- DROP: digunakan untuk menghapus *database*, tabel, atau indeks.

Mempersiapkan Database dan Tabel

Sebelum Anda menginjak ke materi selanjutnya, Anda perlu mempersiapkan dulu *database* dan tabel yang akan digunakan sebagai sarana praktikum. Pada contoh ini, kita akan membuat sebuah *database* dan sebuah tabel.

Praktikum 13-1: Membuat *database* dan tabel

- ☑ Jalankan program MySQL Client
- ☑ Tuliskan sintaks SQL berikut untuk membuat *database* dengan nama SIPEGDB

create database SIPEGDB;

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -p
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.24a-community-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database SIPEGDB;
Query OK, 1 row affected (0.06 sec)

mysql>
```

- ☑ Aktifkan *database* SIPEGDB dengan menuliskan perintah berikut:

use SIPEGDB;

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -p
Database changed
```

- ☑ Tuliskan sintaks SQL berikut untuk membuat tabel PEGAWAI

**create table PEGAWAI (
nip char(6) not null primary key,
nama varchar(25) not null,
alamat varchar(50),
nohp varchar(15)
);**

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -p
Database changed
mysql> create table PEGAWAI (
->   nip char(6) not null primary key,
->   nama varchar(25) not null,
->   alamat varchar(50),
->   nohp varchar(15)
-> );
Query OK, 0 rows affected (0.50 sec)

mysql>
```

- ☑ Isikan beberapa data contoh (*dummy*) menggunakan sintaks SQL berikut:

**insert into pegawai values('PR-001', 'Rani',
'Sriwijaya 88 Bandung', '085620039654');
insert into pegawai values('PR-002', 'Aas Asyanto',
'Mekar Wangi 13 Bandung', '02276669169');
insert into pegawai values('PR-003', 'Ian Raharjo',
'Kembar Siam 711 Bandung', '08132300034');**

```
C:\WINDOWS\system32\cmd.exe mysql root@p
mysql> show tables;
+-----+
| tables_in_pegawai |
+-----+
| pegawai            |
+-----+
mysql> select * from pegawai;
+-----+-----+-----+
| id_pegawai | nama_pegawai | alamat |
+-----+-----+-----+
| 1000000000000000000 | 1000000000000000000 | 1000000000000000000 |
+-----+-----+-----+
mysql> insert into pegawai values('PR-001', 'Ian Raharjo', 'Kediri');
Query OK, 1 row affected (0.34 sec)
mysql>
```

- ☑ Tutup program MySQL Client dengan menulis perintah EXIT.

Keterangan:

Dengan melakukan praktikum di atas, Anda telah memiliki sebuah database dan tabel yang berisi beberapa data dummy. Database dan tabel tersebut akan digunakan sebagai contoh pada praktikum-praktikum selanjutnya.

Membangun Koneksi antara PHP dan MySQL

Pada saat bekerja dengan server MySQL (atau RDBMS lain) di dalam PHP, Anda harus menggunakan tahap-tahap berikut:

1. Buka koneksi ke server database
2. Bekerja dengan data di dalam server database
3. Tutup koneksi

PHP menyediakan fungsi `mysqli_connect()` untuk membangun koneksi antara PHP dan MySQL. Bentuk umum fungsi tersebut adalah:

```
$conn = mysqli_connect(hostname, username, password, database name)
```

Variabel `$conn` adalah variabel yang digunakan sebagai indikator sukses atau tidaknya koneksi database yang dibangun. Jika koneksi gagal, variabel ini akan bernilai `false`.

Contoh:

```
$conn = mysqli_connect("localhost", "pras", "s3cr3T",
                        "sipegdb");

if ($conn) {
    echo "Koneksi sukses.";
    // perintah-perintah yang akan dilakukan
    // ...
} else {
    die("Error: " . mysqli_connect_error());
}
```

Fungsi `mysqli_connect_error()` akan mengembalikan *string* yang merupakan pesan kesalahan yang terjadi pada saat Anda memanggil fungsi `mysqli_connect()`. Fungsi lain yang dapat digunakan untuk keperluan semacam ini adalah `mysqli_connect_errno()`. Perbedaannya, `mysqli_connect_errno()` hanya akan mengembalikan kode *error*-nya saja, bukan berupa *string*.

Menampilkan Data ke Halaman Web

Setelah koneksi berhasil dibangun, langkah selanjutnya yang perlu Anda lakukan adalah mengakses data di dalam database, apakah itu untuk dimanipulasi atau hanya sekadar ditampilkan saja ke dalam halaman web. Pada bagian ini kita akan mempraktikkan cara mengakses data dari database dan menampilkannya ke dalam halaman web.

Terdapat beberapa langkah yang diperlukan untuk menampilkan data dari suatu tabel di dalam database ke halaman web:

1. Buat koneksi ke server MySQL menggunakan fungsi `mysqli_connect()`.
2. Jalankan *query* (dalam hal ini statemen `SELECT`) menggunakan fungsi `mysqli_query()`. Proses ini akan menghasilkan suatu *result set* (kumpulan *record*). Tampunglah *result set* tersebut ke dalam variabel bantu.
3. Gunakan pengulangan untuk menangkap setiap baris data dari *result set* yang dihasilkan. Untuk keperluan ini Anda dapat menggunakan fungsi `mysqli_fetch_row()`. Untuk mempermudah proses, tampunglah baris data yang ditangkap tersebut ke dalam variabel bantu.
4. Melalui variabel bantu tersebut, Anda dapat menampilkan setiap kolom (*field*) yang ada dari baris yang ditangkap
5. Setelah semua baris diproses, bebaskan *result set* dari memori menggunakan fungsi `mysqli_free_result()`.
6. Tutup koneksi ke server.

Langkah-langkah di atas dapat direpresentasikan melalui kode berikut:

```
<?php
$conn = mysqli_connect("localhost", "pras", "s3cr3T",
                      "sipegdb");

if (mysqli_connect_errno()) {
    echo "Koneksi ke server gagal dilakukan.";
    exit();
}

$query = "select nip, nama, alamat, nohp " .
        "from pegawai order by nip";
$result = mysqli_query($conn, $query);

if ($result) {
    while ($row = mysqli_fetch_row($result)) {
        echo "$row[0], $row[1], " .
            "$row[2] ($row[3])<br />";
    }
}
```

```
mysqli_free_result($result);

mysqli_close($conn);
}
```

Fungsi `mysqli_query()` digunakan untuk mengeksekusi *query* yang ditampung dalam variabel `$query`. Pelaksanaan *query* akan dilakukan pada *database* yang telah terhubung, yang direpresentasikan oleh variabel `$conn`. Pemanggilan `mysqli_query()` ini akan menghasilkan suatu *result set*, yang selanjutnya ditampung ke dalam variabel `$result`.

```
$result = mysqli_query($conn, $query);
```

Jika variabel `$result` tidak kosong, program akan menangkap baris demi baris data yang terdapat pada variabel `$result`. Setiap baris yang ditangkap akan disimpan ke dalam variabel `$row`. Proses penangkapan setiap baris data dapat dilakukan menggunakan `mysqli_fetch_row()`. Fungsi ini akan mengembalikan nilai berupa *array*. Jadi, variabel `$row` adalah *array*.

```
$row = mysqli_fetch_row($result)
```

Dari variabel `$row` inilah kita dapat menampilkan data ke dalam halaman web. Dalam contoh kode ini, kita menggunakan perintah `echo` untuk menampilkan data. Variabel `$row` merupakan *array*, yang indeksnya sama urutannya sama seperti urutan *field* pada tabel pegawai. Dengan demikian, kita dapat mengakses setiap nilai yang terkandung dalam variabel `$row` menggunakan perintah berikut:

```
printf "$row[0], $row[1], " .
      "$row[2] ($row[3])<br />";
```

Nilai 0 pada kode di atas merupakan indeks *array* yang merepresentasikan *field* pertama (NIP) pada tabel pegawai, 1 merepresentasikan *field* kedua (NAMA), dan seterusnya.

Catatan:

Dalam menggunakan fungsi `mysqli_connect()`, gantilah nama user dan password sesuai setting yang ada di komputer Anda.

Praktikum 13-2: Menampilkan data dari database ke halaman web

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<html>
<head>
  <title>Menampilkan Data</title>
</head>
<body>
<h2>Demo menampilkan data dari database</h2>
<?php

$dbname = "sipegdb";
$host = "localhost";
$username = "pras";
$password = "s3cr3T";

$conn = mysqli_connect($host, $username, $password,
                      $dbname);

if (mysqli_connect_errno()) {
  echo "Koneksi ke server gagal dilakukan.";
  exit();
}

$query = "select nip, nama, alamat, nohp "
        "from pegawai order by nip";
$result = mysqli_query($conn, $query);

if ($result) {
  ?>
  <h4><u>Daftar Pegawai PT. Multidimension:</u></h4>
  <table border="1">
    <tr>
      <th width="80">NIP</th>
      <th width="110">Nama</th>
```

```
      <th width="200">Alamat</th>
      <th width="130">No. HP</th>
    </tr>
    <?php
    while ($row = mysqli_fetch_row($result)) {
      ?>
      <tr>
        <?php
        $nip = $row[0];
        $nama = $row[1];
        $alamat = $row[2];
        $nohp = $row[3];
        ?>
        <td><?php echo $nip;?></td>
        <td><?php echo $nama;?></td>
        <td><?php echo $alamat;?></td>
        <td><?php echo $nohp;?></td>
      </tr>
    <?php
  }
  ?>
</table>
<?php
mysqli_free_result($result);
}

mysqli_close($conn);
?>
</body>
</html>
```

- ☒ Simpan file dengan nama **showdata.php**
- ☒ Jalankan web browser dan masukkan **<http://localhost/modul-PHP/showdata.php>**

Keterangan:

Pada praktikum kali ini kita memadukan kode PHP dengan HTML. Kode HTML yang terdapat di dalam contoh di atas digunakan untuk membuat tabel. Adapun data-data yang ditampilkan di dalam tabel merupakan data yang diambil dari database MySQL.

Menampilkan Data - Studio Function

File Edit View History Bookmarks Tools Help

http://localhost/jarak1190/showdata.php

Getting Started Latest Headlines

Demo menampilkan data dari database

Daftar Pegawai PT. Multidimension:

NIP	Nama	Alamat	No. HP
PR-001	Rani	Sriwijaya 88 Bandung	085620039654
PR-002	Aas Asyanto	Mekar Wangi 13 Bandung	02276669169
PR-003	Ian Raharjo	Kembar Siam 711 Bandung	08132300034

Done

Seperti yang Anda lihat, mula-mula kita membuat koneksi ke server MySQL. Jika sukses, langkah selanjutnya yang dilakukan adalah melakukan query terhadap tabel pegawai, dan hasilnya (berupa result set) akan disimpan di dalam variabel `$result`.

```
$query = "select nip, nama, alamat, nohp " .  
        "from pegawai order by nip";  
$result = mysqli_query($conn, $query);
```

Untuk memulai pembuatan tabel, kita perlu menyisipkan kode HTML ke dalam kode di atas. Mula-mula kita membentuk satu baris paling atas, yang akan digunakan sebagai header atau judul masing-masing kolom pada tabel bersangkutan. Untuk keperluan ini, kita dapat menggunakan tag `<th>` dan `</th>`, seperti berikut:

```
<tr>  
  <th width="80">NIP</th>  
  <th width="110">Nama</th>  
  <th width="200">Alamat</th>  
  <th width="130">No. HP</th>  
</tr>
```

Setelah itu, kita perlu menangkap setiap baris data yang tersimpan di dalam variabel `$result`. Kode yang kita gunakan di sini adalah:

```
  <?php  
  while ($row = mysqli_fetch_row($result)) {  
    <tr>  
      <?php  
      $nip = $row[0];  
      $nama = $row[1];  
      $alamat = $row[2];  
      $nohp = $row[3];  
      ?>  
      <td><?php echo $nip;?></td>  
      <td><?php echo $nama;?></td>  
      <td><?php echo $alamat;?></td>  
      <td><?php echo $nohp;?></td>  
    </tr>  
  <?php  
  }  
  ?>
```

Melalui kode tersebut, program akan menangkap setiap baris data dan menyimpannya ke dalam variabel `$row`. Tag `<td>` berguna untuk memasukkan data yang ditangkap ke dalam bentuk tabel. Proses ini akan diulang sebanyak jumlah baris yang terdapat pada tabel `PEGAWAI`.

Memasukkan Data Melalui Form

Pada praktiknya, proses memasukkan data ke dalam database dilakukan melalui form, bukan dimanipulasi langsung melalui MySQL Client. Contoh yang sering kita jumpai adalah seperti pada proses registrasi *e-mail*. Pada proses tersebut, kita disuguhi form registrasi dan diminta untuk memasukkan data-data pribadi kita. Data-data tersebut selanjutnya akan disimpan ke dalam database server yang mereka kelola.

Pada bagian ini, kita akan mempraktikkan cara membuat *form* (user interface) pada halaman *web* yang dapat berinteraksi dengan data di dalam *database server*.

Sintaks SQL yang digunakan untuk memasukkan data adalah:

```
insert into NamaTabel(field1, field2, ...) values(nilai1, nilai2, ...)
```

Contoh:

```
insert into pegawai(nip, nama, alamat, nohp)
values('PR-004', 'Dadan Sukandar',
      'Sarijadi 234 Bandung', '081322762341');
```

Praktikum 13-3: Memasukkan data ke *database* melalui *form*

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```
<html>
<head>
  <title>Memasukkan Data</title>
</head>
<body>
<h2>Demo memasukkan data ke dalam database</h2>

<form action="insert.php" method="post">
<strong>NIP:</strong><br />
<input name="nip" type="text" size="10" maxlength="6">
<br /><br />
<strong>Nama:</strong><br />
<input name="nama" type="text" size="30" maxlength="25">
<br /><br />
<strong>Alamat:</strong><br />
<input name="alamat" type="text" size="50"
maxlength="50">
<br /><br />
<strong>No. HP:</strong><br />
<input name="nohp" type="text" size="15" maxlength="15">
<br /><br />
```

```
<input type="submit" name="btnSubmit" value="Simpan">
</form>

</body>
</html>
```

- ☑ Simpan *file* dengan nama **form-insert.html**
- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```
<?php
if (isset($_POST['btnSubmit'])) {
  $nip = $_POST['nip'];
  $nama = $_POST['nama'];
  $alamat = $_POST['alamat'];
  $nohp = $_POST['nohp'];

  $conn = mysqli_connect("localhost", "pras",
                        "s3cr3T", "sipegdb");

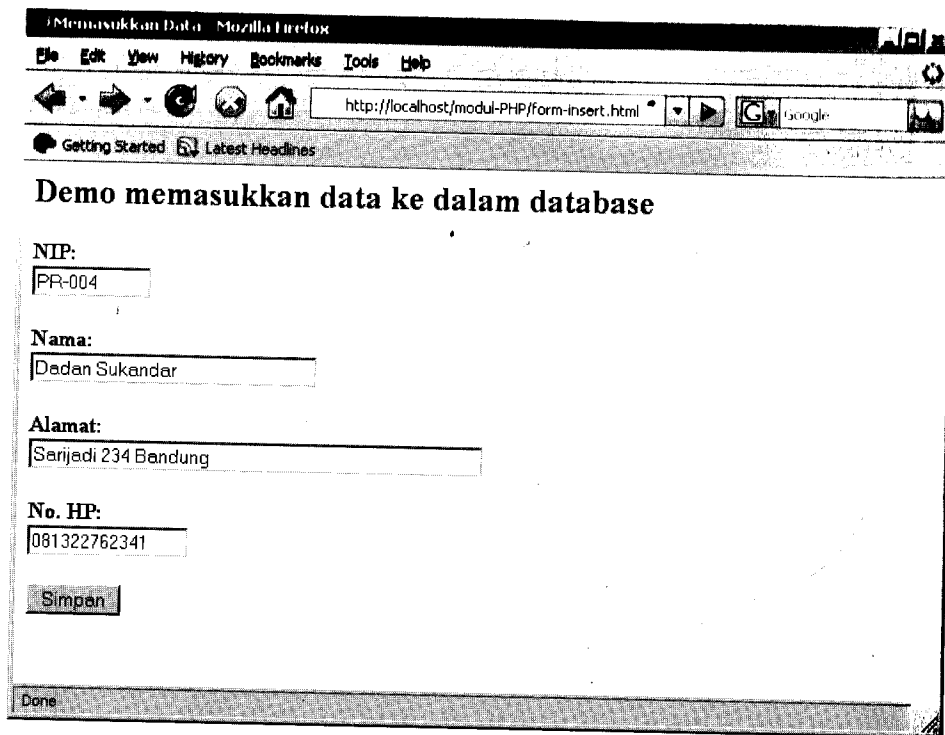
  if (mysqli_connect_errno()) {
    echo "Koneksi ke server gagal";
    exit();
  }

  $sql = "insert into pegawai(nip, nama, alamat, nohp) " .
        "values('$nip', '$nama', '$alamat', '$nohp')";

  mysqli_query($conn, $sql);
  $num = mysqli_affected_rows();

  if ($num > 0) {
    echo "Data yang Anda masukkan sudah disimpan.";
  } else {
    echo "Data gagal disimpan ke dalam database";
  }
}
?>
```

- ☑ Simpan *file* dengan nama **insert.php**
- ☑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/form-insert.php**

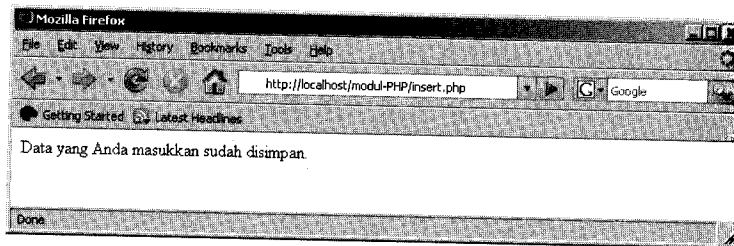


- ☑ Isikan data-data yang diminta pada *form* di atas, lalu klik tombol **Simpan**.

Keterangan:

Pada praktikum kali ini, kita menggunakan `mysqli_query`

(`)` untuk mengeksekusi statemen `INSERT`.



```
$sql = "insert into pegawai(nip, nama, alamat, nohp) "
      "values('$nip', '$nama', '$alamat', '$nohp')";
mysqli_query($conn, $sql);
```

Pada saat suatu tabel telah dimodifikasi (ditambah, diubah, maupun dihapus datanya), jumlah baris data yang dimodifikasi pada tabel tersebut dapat kita peroleh dengan memanggil fungsi `mysqli_affected_rows()`.

```
$num = mysqli_affected_rows();
```

Melalui kode di atas, variabel `$num` akan menyimpan jumlah baris yang baru dimasukkan ke dalam tabel pegawai. Jika statemen `INSERT`, `UPDATE`, atau `DELETE` yang dilaksanakan menggunakan fungsi `mysqli_query()` ternyata salah (terdapat kesalahan sintaks SQL atau kesalahan lain), maka fungsi `mysqli_affected_rows()` akan mengembalikan nilai `-1`. Dengan demikian, indikator sukses atau tidaknya proses `INSERT` yang kita lakukan ditentukan oleh nilai dari variabel `$num`. Jika nilainya lebih besar dari nol, maka data berhasil dimasukkan ke dalam tabel `PEGAWAI`. Jika sebaliknya, berarti data gagal disimpan ke database. Pemeriksaan nilai variabel `$num` dapat kita lakukan melalui kode berikut:

```
if ($num > 0) {
    echo "Data yang Anda masukkan sudah disimpan.";
} else {
    echo "Data gagal disimpan ke dalam database";
}
```

Mengubah Data dari Form Web

Perubahan data pada suatu tabel di dalam *database* dapat dilakukan dengan mengirimkan perintah `UPDATE` menggunakan `mysqli_query()`. Karena dalam contoh ini kita hanya akan melakukan perubahan satu baris data, maka kita perlu menentukan baris mana yang akan diubah. *Field* penentu yang akan kita gunakan di sini adalah *field* `NIP`, karena *field* tersebut merupakan *primary key* dalam tabel `PEGAWAI`.

Untuk keperluan semacam ini, kita akan memodifikasi halaman *web* yang menampilkan data pegawai. Tujuannya adalah agar setiap baris data yang ditampilkan memiliki *link* ke *form* perubahan. Dalam *link* tersebut, kita dapat mengirimkan parameter *nip* melalui URL tujuan yang akan dilewatkan.

Praktikum 13-4: Mengubah data melalui *form*

- ☑ Jalankan aplikasi Text Editor
- ☑ Buka file **showdata.php**, lalu ubah kodenya menjadi seperti berikut:

```
<html>
<head>
  <title>Mengubah Data</title>
</head>
<body>
  <h2>Demo mengubah data</h2>
  <?php

  $dbname = "sipegdb";
  $host = "localhost";
  $username = "pras";
  $password = "s3cr3T";

  $conn = mysqli_connect($host, $username, $password,
                        $dbname);

  if (mysqli_connect_errno()) {
    echo "Koneksi ke server gagal dilakukan.";
    exit();
  }

  $query = "select nip, nama, alamat, nohp " .
    "from pegawai order by nip";
  $result = mysqli_query($conn, $query);

  if ($result) {
    ?>
    <h4><u>Daftar Pegawai PT. Multidimension:</u></h4>
    <table border="1">
      <tr>
        <th width="80">NIP</th>
```

```
        <th width="110">Nama</th>
        <th width="200">Alamat</th>
        <th width="130">No. HP</th>
      </tr>
      <?php
      while ($row = mysqli_fetch_row($result)) {
        ?>
        <tr>
          <?php
          $nip = $row[0];
          $nama = $row[1];
          $alamat = $row[2];
          $nohp = $row[3];
          ?>
          <td><?php echo $nip;?></td>
          <td>
            <a href="form-update.php?nip=<?php
              echo $nip;?>">
              <?php echo $nama;?></a></td>
          <td><?php echo $alamat;?></td>
          <td><?php echo $nohp;?></td>
        </tr>
      <?php
    }
    ?>
  </table>
  <br />
  <strong>Keterangan: </strong>
  Klik link yang tersedia untuk mengubah data!
  <?php
  mysqli_free_result($result);
}

mysqli_close($conn);
?>
</body>
</html>
```

- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```
<html>
<head>
  <title>Mengubah Data</title>
</head>
```



```
<body>
```

```
<h2>Form Perubahan Data</h2>
```

```
<?php
```

```
$nip = $_GET['nip'];
```

```
$conn = mysqli_connect("localhost", "pras",  
"s3cr3T", "sipegdb");
```

```
if (mysqli_connect_errno()) {  
    echo "Koneksi ke server gagal";  
    exit();  
}
```

```
$query = "select nama, alamat, nohp from pegawai "  
"where nip = '$nip'";
```

```
$result = mysqli_query($conn, $query);
```

```
if ($result) {
```

```
    $row = mysqli_fetch_array($result);
```

```
}
```

```
?>
```

```
<form action="update.php" method="post">
```

```
<strong>NIP:</strong><br />
```

```
<?php echo $nip; ?>
```

```
<input name="nip" type="hidden"
```

```
value="<?php echo $nip; ?>">
```

```
<br /><br />
```

```
<strong>Nama:</strong><br />
```

```
<input name="nama" type="text" size="30" maxlength="25"
```

```
value="<?php echo $row[0]; ?>">
```

```
<br /><br />
```

```
<strong>Alamat:</strong><br />
```

```
<input name="alamat" type="text" size="50"
```

```
maxlength="50" value="<?php echo $row[1]; ?>">
```

```
<br /><br />
```

```
<strong>No. HP:</strong><br />
```

```
<input name="nohp" type="text" size="15" maxlength="15"
```

```
value="<?php echo $row[2]; ?>">
```

```
<br /><br />
```

```
<input type="submit" name="btnSubmit" value="Simpan">
```

```
</form>
```

```
</body>
```

```
</html>
```

❑ Simpan file dengan nama **form-update.php**

❑ Buatlah kembali file baru, dan tuliskan kode berikut:

```
<?php
```

```
if (isset($_POST['btnSubmit'])) {
```

```
    $nip = $_POST['nip'];
```

```
    $nama = $_POST['nama'];
```

```
    $alamat = $_POST['alamat'];
```

```
    $nohp = $_POST['nohp'];
```

```
$conn = mysqli_connect("localhost", "pras",  
"s3cr3T", "sipegdb");
```

```
if (mysqli_connect_errno()) {
```

```
    echo "Koneksi ke server gagal";
```

```
    exit();  
}
```

```
$sql = "update pegawai "
```

```
"set nama = '$nama', "
```

```
"alamat = '$alamat', "
```

```
"nohp = '$nohp' "
```

```
"where nip = '$nip'";
```

```
mysqli_query($conn, $sql);
```

```
$num = mysqli_affected_rows($conn);
```

```
if ($num > 0) {
```

```
    echo "Perubahan data telah disimpan.";
```

```
?><br /><br />
```

```
<a href="showdata.php">Lihat perubahan</a><?php
```

```
} else {
```

```
    echo "Perubahan data gagal disimpan.";
```

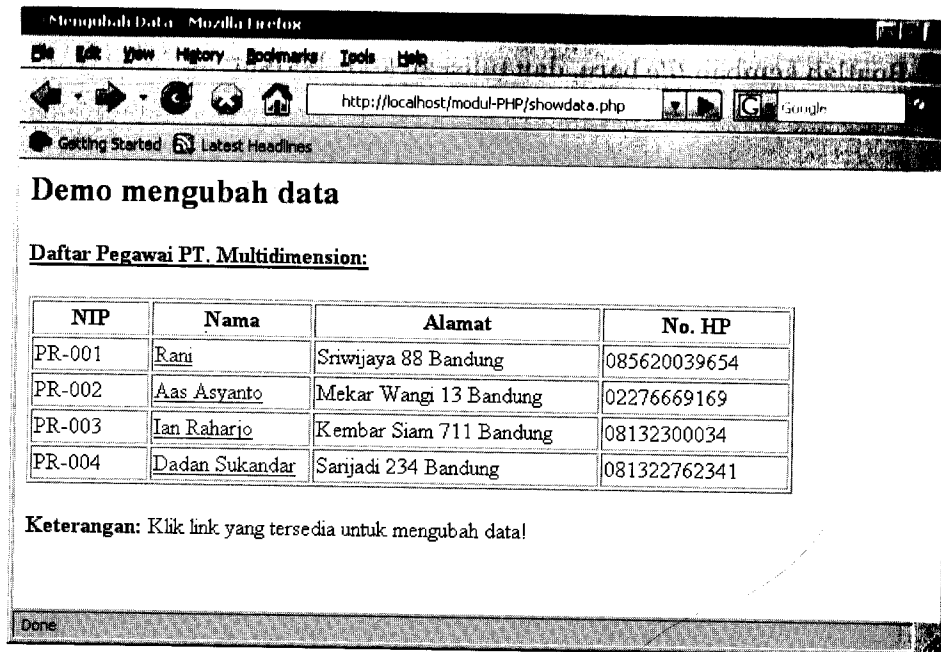
```
}
```

```
}
```

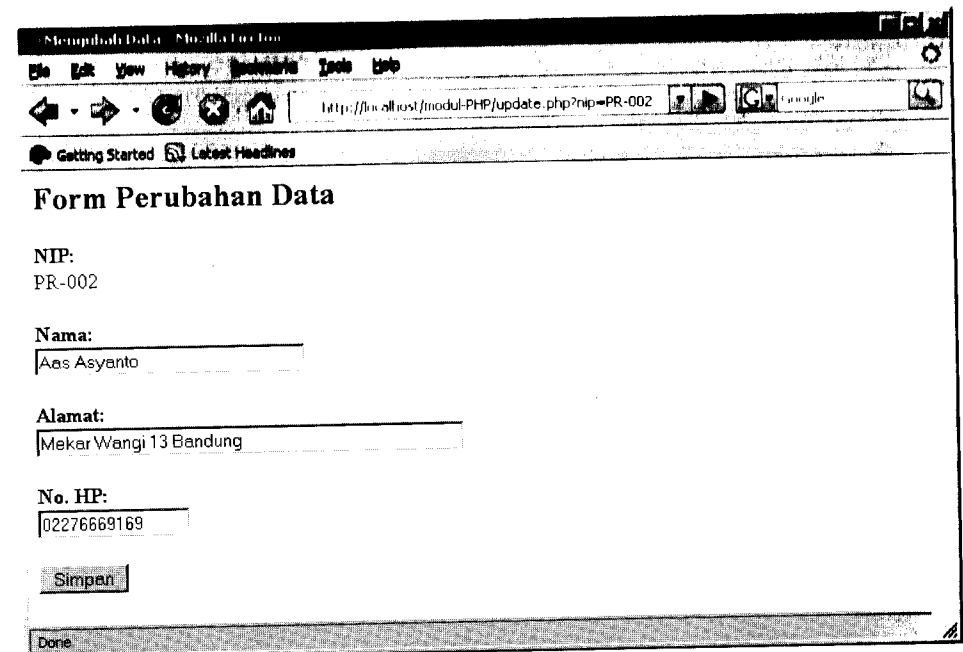
```
?>
```

❑ Simpan file dengan nama **update.php**

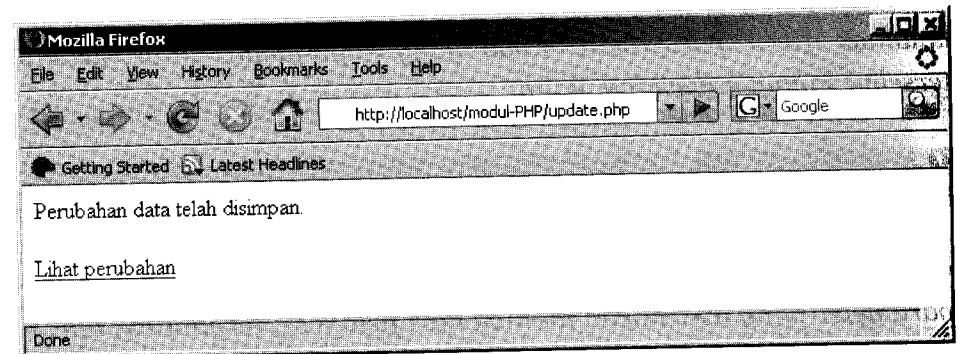
❑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/showdata.php**



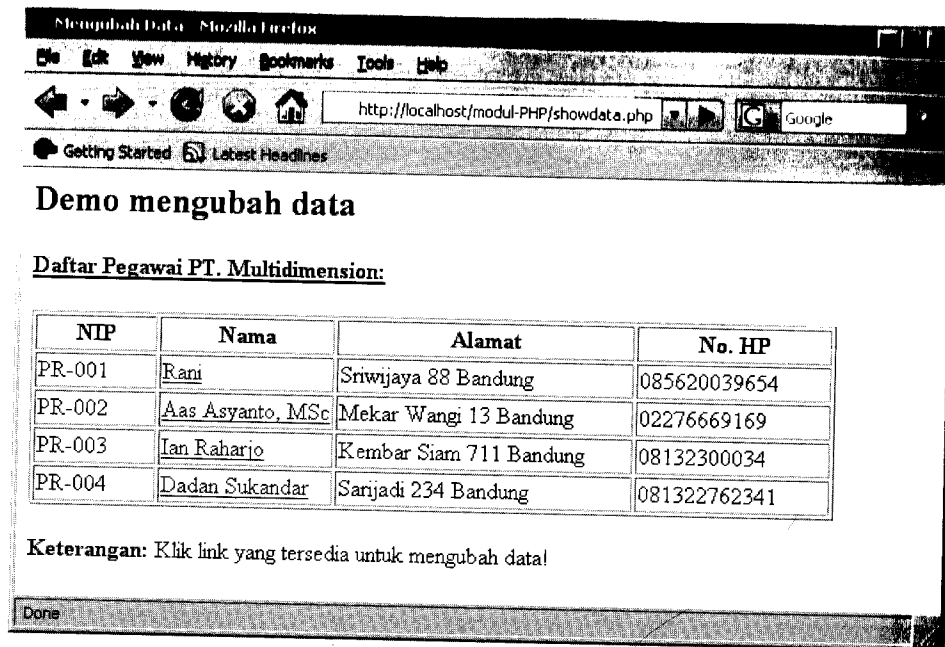
- ☑ Klik salah satu baris data untuk menampilkan *form* perubahan data.



- ☑ Lakukan perubahan data (misalnya dari **Aas Asyanto** menjadi **Aas Asyanto, MSc**), lalu klik tombol **Simpan**



- ☑ Klik *link* **Lihat perubahan** untuk melihat perubahan data yang telah dilakukan oleh program



Keterangan:

Pada praktikum kali ini kita melakukan perubahan kecil terhadap file **showdata.php** untuk membuat data nama yang tampil menjadi link ke halaman form perubahan. Kode yang diperlukan untuk kebutuhan tersebut sangat sederhana, yaitu:

```
<a href="form-update.php?nip=<?php echo $nip;?>">
<?php echo $nama;?></a>
```

Dalam kode di atas kita membuat parameter dengan nama **nip**. Nilai parameter yang dikirimkan diambil dari field **NIP**, tergantung dari baris mana yang user pilih. Dengan kata lain, setiap baris data yang tampil di atas akan mengirimkan nilai parameter **nip** yang berbeda-beda.

Parameter yang dikirimkan oleh file **showdata.php** akan ditangkap di dalam file **form-update.php**. Cara menangkapnya adalah dengan menggunakan kode berikut:

```
$nip = $_GET['nip'];
```

Parameter tersebut berfungsi untuk menampilkan data yang dipilih ke dalam elemen kontrol (**TEXTFIELD**) yang disediakan dalam form perubahan data.

Setelah user melakukan perubahan data yang diinginkan dan memilih tombol **SUBMIT**, maka program akan mengeksekusi kode yang terdapat pada file **update.php**. Dalam kode tersebut, kita menggunakan statemen **UPDATE** untuk mengubah data lama dengan data baru. Agar parameter **nip** tetap dikenali oleh file **update.php**, di sini kita menggunakan elemen **HIDDEN CONTROL** di dalam file **form-update.php**.

```
<input name="nip" type="hidden"
value="<?php echo $nip;?>">
```

Karena form yang terdapat pada file **form-update.php** menggunakan metode **POST**, maka cara menangkap nilai **nip** di dalam file **update.php** adalah:

```
$nip = $_POST['nip'];
```

Menghapus Data dari Form Web

Proses penghapusan data di dalam database dilakukan dengan menggunakan statemen **DELETE**. Bentuk umum dari perintah **DELETE** adalah:

```
delete from NamaTabel where kondisi
```

Contoh:

```
delete from pegawai where nip = 'PR-003';
```

Perintah di atas digunakan untuk menghapus data dalam tabel **PEGAWAI** yang kolom **NIP**-nya bernilai 'PR-003';

Pada bagian ini, kita akan melakukan modifikasi ulang *file* **showdata.php** untuk mendemonstrasikan proses penghapusan data. Dalam contoh ini, kita akan mengizinkan *user* untuk memilih data-data mana saja yang ingin dihapus, dengan cara menyertakan elemen kontrol **CHECK BOX** pada setiap baris data yang ditampilkan.

Praktikum 13-4: Menghapus data melalui form

- ☑ Jalankan aplikasi Text Editor
- ☑ Buka kembali *file* **showdata.php**, lalu ubah kodenya menjadi seperti berikut:

```
<html>
<head>
  <title>Menghapus Data</title>
</head>
<body>
<h2>Demo menghapus data</h2>
<?php

$dbname = "sipegdb";
$host = "localhost";
$username = "pras";
$password = "s3cr3T";

$conn = mysqli_connect($host, $username, $password,
                      $dbname);

if (mysqli_connect_errno()) {
  echo "Koneksi ke server gagal dilakukan.";
  exit();
}

$query = "select nip, nama, alamat, nohp "
        "from pegawai order by nip";
$result = mysqli_query($conn, $query);
```

```
if ($result) {
  ?>
  <h4><u>Daftar Pegawai PT. Multidimension:</u></h4>
  <form action="delete.php" method="post">
  <table border="1">
    <tr>
      <th></th>
      <th width="80">NIP</th>
      <th width="110">>Nama</th>
      <th width="200">Alamat</th>
      <th width="130">No. HP</th>
    </tr>
    <?php
    while ($row = mysqli_fetch_row($result)) {
      ?>
      <tr>
        <?php
        $nip = $row[0];
        $nama = $row[1];
        $alamat = $row[2];
        $nohp = $row[3];
        ?>
        <td>
          <input name="data[]" type="checkbox"
            Value="<?php echo $nip; ?>">
        </td>
        <td><?php echo $nip;?></td>
        <td>
          <a href="form-update.php?nip=<?php
            echo $nip;?>">
            <?php echo $nama;?></a></td>
        <td><?php echo $alamat;?></td>
        <td><?php echo $nohp;?></td>
      </tr>
    <?php
    }
  ?>
  </table>
  <br />
  <strong>Keterangan: </strong>
  Pilih data yang akan dihapus dengan memberi tanda
  cek pada kotak yang tersedia!
  <br /><br />
  <input type="submit" name="btnSubmit" value="Hapus">
```



```

</form>
<?php
mysql_free_result($result);
}

mysql_close($conn);
?>
</body>
</html>

```

- ☑ Buatlah *file* baru, dan tuliskan kode berikut:

```

<?php
if (isset($_POST['btnSubmit'])) {
    $data = $_POST['data'];

    $conn = mysqli_connect("localhost", "pras",
        "s3cr3T", "sipegdb");
    if (mysqli_connect_errno()) {
        echo "Koneksi ke server gagal";
        exit();
    }

    for ($i=0; $i < sizeof($data); $i++) {
        $sql = "delete from pegawai "
            . "where nip = '$data[$i]'";

        mysqli_query($conn, $sql);
    }

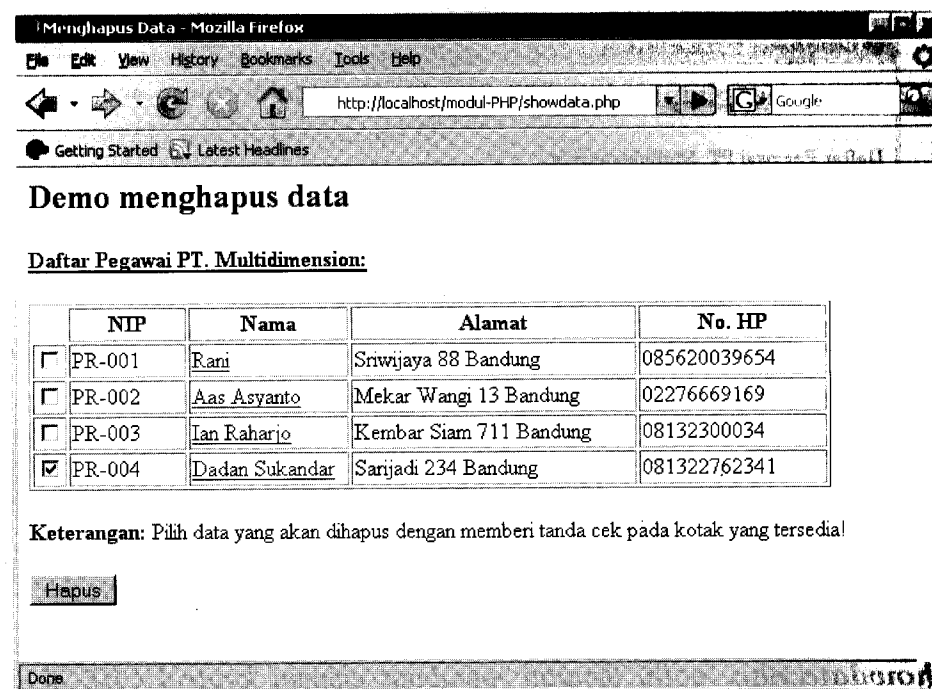
    $num = mysqli_affected_rows($conn);

    if ($num > 0) {
        echo " $num baris data telah dihapus.";
        ?><br /><br />
        <a href="showdata.php">Lihat perubahan</a><?php
    } else {
        echo "Penghapusan data gagal dilakukan.";
    }
}
?>

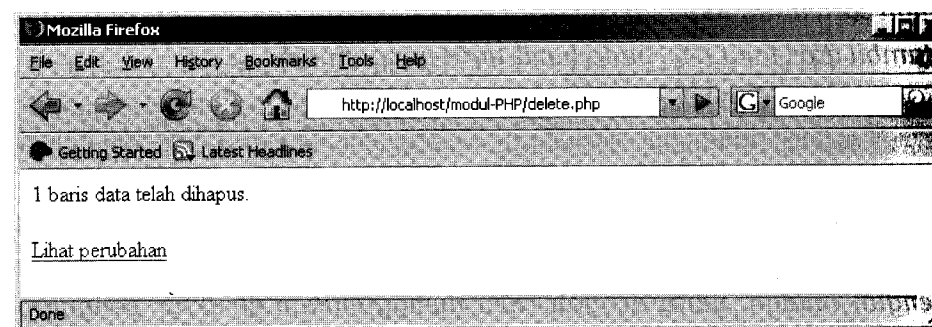
```

- ☑ Simpan *file* dengan nama **delete.php**

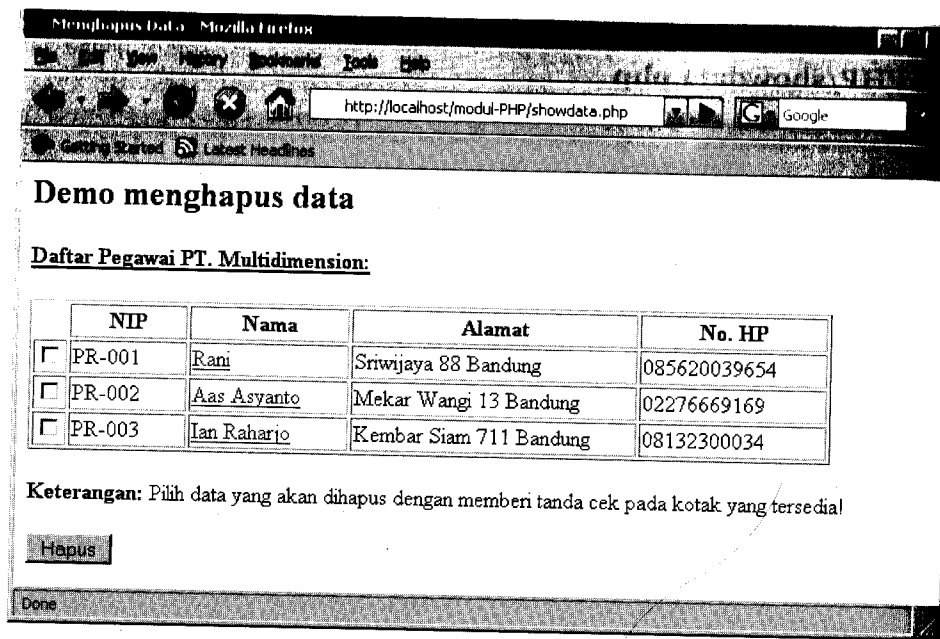
- ☑ Jalankan *web browser* dan masukkan **http://localhost/modul-PHP/showdata.php**



- ☑ Pilih data yang akan dihapus, lalu klik tombol **Hapus**



- ☑ Untuk melihat perubahan, klik *link* **Lihat perubahan**



Keterangan:

Pada praktikum kali ini kita menambah sedikit kode ke dalam file **showdata.php**, yaitu dengan membuat elemen kontrol CHECK BOX pada setiap baris yang ditampilkan. Elemen kontrol tersebut kita tampung ke dalam sebuah array, dengan nama `$data`. Jika user memilih salah satu data, maka array `$data` akan bertambah elemen, dan nilainya diambil dari field NIP pada baris data yang dipilih.

```
<input name="data[]" type="checkbox"
Value="<?php echo $nip; ?>">
```

Jadi, array `$data` berisi kumpulan NIP dari baris data yang akan dihapus. Dengan demikian, proses penghapusan data dapat kita lakukan dengan mudah, yaitu dengan cara menggunakan struktur pengulangan pada setiap elemen array. Kode yang kita gunakan di sini adalah:

```
for ($i=0; $i < sizeof($data); $i++) {
    $sql = "delete from pegawai ".
```

```
"where nip = '$nip'";
```

```
mysqli_query($conn, $sql);
```

Jumlah baris yang dihapus dapat kita ketahui dengan cara memanggil fungsi `mysqli_affected_rows()`.

```
$num = mysqli_affected_rows($conn);
```

Jika nilai `$num` lebih besar nol maka proses penghapusan data berhasil dilakukan. Jika sebaliknya, berarti penghapusan data gagal. Pemeriksaan nilai `$num` dapat dilakukan menggunakan kode berikut:

```
if ($num > 0) {
    echo "$num baris data telah dihapus.";
    ?><br /><br />
    <a href="showdata.php">Lihat perubahan</a><?php
} else {
    echo "Penghapusan data gagal dilakukan.";
}
```


Pengaturan Session

Untuk *website* yang memiliki otentikasi *user* (terdapat *form login* untuk pemakainya), kita perlu mengidentifikasi dan melakukan pemantauan terhadap kegiatan *user* yang sedang aktif atau telah melakukan *login*. Untuk memenuhi kebutuhan semacam ini, PHP menyediakan suatu mekanisme yang disebut *session*.

Dalam modul ini Anda akan mempelajari materi-materi berikut:

- Mengetahui sistem pengaturan session dalam PHP
- Melakukan inisialisasi sebelum menggunakan session
- Menjalankan dan mendaftarkan session

- Mengetahui isi dari *file session*
- Melacak nilai variabel dari halaman yang satu ke halaman lainnya melalui session
- Menghapus variabel dari suatu session
- Mengakhiri session

Mengenal Sistem Pengaturan Session dalam PHP

Secara *default*, setelah *web server* selesai melayani permintaan user, koneksi antara klien dan *server* akan terputus. Dengan demikian, jika terdapat seorang *user* yang mengakses halaman *web* yang sama sebanyak 10 kali, *web server* akan menganggapnya sebagai 10 *user* (10 pengunjung).

Pengaturan session mengizinkan kita untuk melacak variabel yang berasosiasi dengan seorang user tertentu dalam suatu *web session*. *Web session* itu sendiri adalah waktu yang *user* gunakan untuk menjelajahi halaman-halaman *web* dalam *website* yang kita kelola menggunakan satu *web browser* yang sama. Jika *web browser* ditutup, variabel session akan hilang (tidak lagi berasosiasi dengan *user*), kecuali kita menyediakan cara khusus yang dapat menyimpan session dan kemudian menghubungkannya kembali dengan *user* bersangkutan.

Setiap *user* akan memiliki ID session yang berbeda-beda. Dengan demikian, kita dapat dengan mudah melacak kegiatan atau aktivitas yang sedang/terlah dilakukan oleh *user* tertentu melalui ID session.

Session biasanya dilacak oleh PHP dengan menempatkan *cookie* di dalam *web browser*. *Cookie* yang dibuat tersebut akan mengacu ke ID session yang telah didaftarkan di *web server*. Namun, cara ini memiliki satu kelemahan karena ada juga *user* yang lebih suka mematikan (*disable*) *cookie* pada *web browser* yang mereka gunakan. Untuk mengatasi masalah

ini, pada umumnya para programmer PHP lebih memilih untuk menyertakan ID session di dalam setiap halaman *web*. Dengan demikian, ID session akan tetap dapat dilacak meskipun *user* mengunjungi beberapa halaman yang berbeda.

Inisialisasi yang Perlu Dilakukan

Sebelum kita menggunakan session dalam kode program yang akan kita buat, kita perlu melakukan konfigurasi pada *file php.ini*, yaitu pada baris berikut:

```
session.save_path =
```

Tentukan direktori tujuan pada baris tersebut, misalnya menjadi seperti berikut:

```
session.save_path = "C:\WINDOWS\Temp"
```

Dengan cara seperti di atas, *web server* akan menciptakan *file* baru di dalam direktori C:\WINDOWS\Temp setiap ada session baru yang dijalankan melalui kode program.

Langkah-langkah Menggunakan Session

Langkah pertama untuk menggunakan session di dalam suatu kode program adalah dengan memanggil fungsi `session_start()`:

```
session_start();
```

Fungsi `session_start()` tidak memerlukan argumen. Jika Anda menjalankan session baru, maka fungsi akan melakukan inisialisasi dan membentuk *file* temporari yang dibutuhkan untuk melacak session tersebut.

Sekali session dijalankan, kita perlu mendaftarkan satu atau lebih variabel ke dalam session tersebut. Session tidak dapat melacak nilai suatu variabel jika variabel tersebut belum didaftarkan. Fungsi yang digunakan untuk keperluan ini adalah `session_register()`, yang memiliki bentuk umum seperti berikut:

```
session_register(String)
```

Argumen `String` adalah nama variabel yang ingin didaftarkan ke dalam session. Melalui cara ini, variabel tersebut akan tetap dikenali oleh halaman lain yang juga mengimplementasikan session.

Praktikum 14-1: Menggunakan session untuk melacak kunjungan user

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<?php
session_start();
$idsession = session_id();
if (!session_is_registered("count")) {
    session_register("count");
    $_SESSION['count'] = 1;
}
?>

<html>
<head>
    <title>Menggunakan Session</title>
</head>
<body>
<h2>Demo menggunakan session (Bagian 1)</h2>

<?php
echo "ID Session: <strong>$idsession</strong><br />";
echo "Anda telah mengunjungi halaman ini sebanyak " .
    $_SESSION['count'] . " kali."
    $_SESSION['count'] = $_SESSION['count'] + 1;
```

?

```
</body>
</html>
```

- ✎ Simpan file dengan nama **session1.php**
- ✎ Jalankan web browser dan masukkan **http://localhost/modul-PHP/session1.php**
- ✎ Lakukan *refresh (reload)* sebanyak 12 kali pada halaman tersebut

Keterangan:

Mula-mula kita menjalankan session baru menggunakan fungsi

`session_start()`. Perlu Anda

perhatikan dan ingat baik-baik bahwa fungsi ini harus ditulis pada bagian paling atas dari setiap kode/file yang Anda buat. Identitas (ID) dari session yang baru saja dijalankan dapat kita ketahui dengan cara memanggil fungsi `session_id()`:

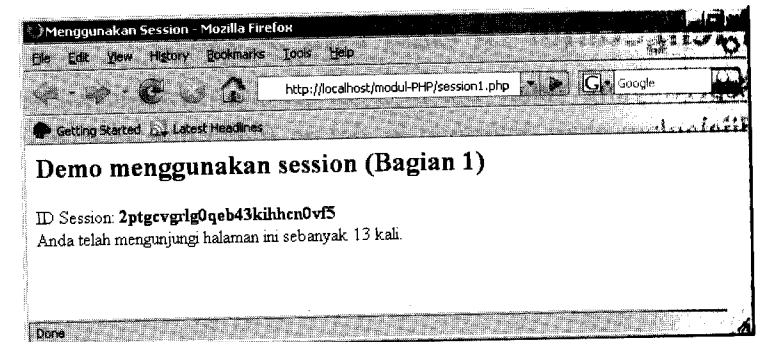
```
$idsession = session_id();
```

Melalui kode tersebut, variabel `$idsession` akan berisi ID session.

Selanjutnya, kita ingin mendaftarkan variabel dengan nama "count" ke dalam session di atas menggunakan kode berikut:

```
session_register("count");
```

Dalam contoh ini, sebelum mendaftarkan variabel "count" kita melakukan pemeriksaan terlebih dahulu, apakah variabel tersebut sudah didaftarkan



atau belum. Untuk melakukan hal ini, kita dapat menggunakan fungsi `session_is_registered()`.

Kita dapat menentukan dan mengakses nilai yang terdapat pada variabel "count" yang telah didaftarkan ke dalam session menggunakan `$_SESSION['count']`.

Mengetahui Isi dari File Session

Seperti yang telah disebutkan sebelumnya bahwa setiap session yang dijalankan akan disimpan ke dalam file khusus. File tersebut selalu diawali dengan nama "sess_" dan diikuti oleh string yang dihasilkan secara otomatis oleh PHP (yang merupakan ID session bersangkutan).

Contoh:

sess_2ptgcvgrlgoqeb43kihcnovf5

File tersebut akan tersimpan di dalam direktori yang telah Anda tentukan pada file php.ini melalui `session.save_path` (dalam contoh ini, C:\WINDOWS\Temp).

Isi dari file tersebut adalah:

count|i:13;

count adalah nama variabel yang didaftarkan ke dalam session, **i** menunjukkan tipe variabel (dalam contoh ini, "i" menunjukkan bahwa variabel **count** bertipe *integer*), dan **13** adalah nilai dari variabel **count**.

Jika Anda mendaftarkan lebih dari satu variabel ke dalam session, maka informasi dari variabel-variabel tersebut juga akan diisikan ke dalam file

sesuai urutan pada saat didaftarkan. Sebagai contoh, jika Anda mendaftarkan variabel **var1** dan **var2**, maka contoh isi file-nya adalah:

var1|i:7;var2|i:21;

Berikut ini daftar simbol yang digunakan untuk merepresentasikan tipe variabel tertentu di dalam file session.

Simbol	Tipe Variabel
i	Integer
s	String
b	Boolean
d	Float
a	Array
O	Objek
N	NULL

Praktikum 14-2: Mengetahui isi dari file session

- ☒ Jalankan aplikasi Text Editor
- ☒ Tuliskan kode berikut:

```
<?php
    session_start();
    $idsession = session_id();
    session_register("nama");
    session_register("usia");
?>

<html>
<head>
    <title>Menggunakan Session</title>
</head>
<body>
<h2>Demo menggunakan session (Bagian 2)</h2>

<?php
    echo "ID Session: <strong>$idsession</strong><br />";
    $_SESSION['nama'] = "Bimo Haryo Hanggoro";
```

```

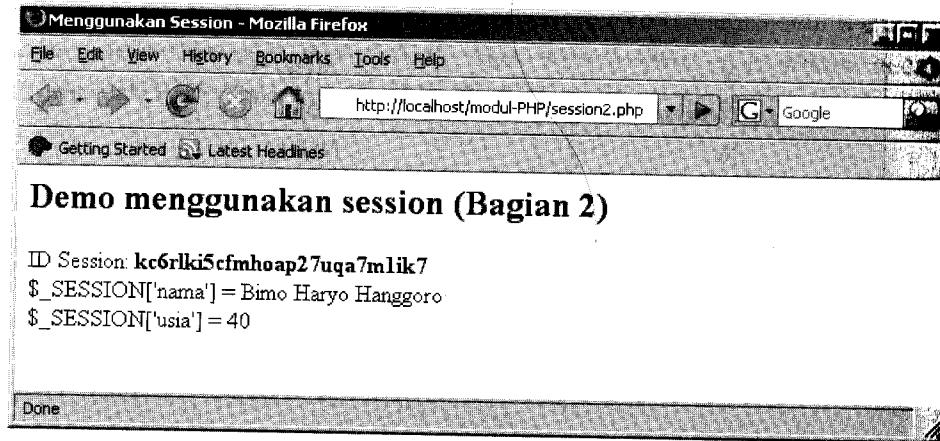
$ _SESSION[ 'usia' ] = 40;

echo "\$_SESSION[ 'nama' ] = " . $_SESSION[ 'nama' ] .
"<br />";
echo "\$_SESSION[ 'usia' ] = " . $_SESSION[ 'usia' ];
?>

</body>
</html>

```

- ☑ Simpan file dengan nama **session2.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/session2.php**



- ☑ Buka file **sess_kc6rlki5cfmhoap27uqa7m1ik7**, Anda akan melihat informasi seperti berikut:

nama|s:19:"Bimo Haryo Hanggoro";usia|i:40;

Keterangan:

Seperti yang Anda lihat pada hasil di atas, kali ini kita mendaftarkan dua buah variabel dengan tipe berbeda: string dan integer. Pada variabel *nama*, simbol yang digunakan adalah "s" yang menunjukkan bahwa nilai dari variabel tersebut bertipe string. Nilai 19 yang ditampilkan di atas

merupakan panjang string atau jumlah karakter yang terdapat pada string bersangkutan.

Melacak Nilai Variabel dari Halaman Lain

Pada bagian ini Anda akan mempraktikkan cara melacak nilai variabel dari halaman-halaman lain yang user kunjungi. Artinya, meskipun user mengunjungi banyak halaman web, nilai variabel tersebut akan tetap dijaga dan dapat diakses dari setiap halaman yang ada. Inilah yang merupakan kegunaan utama session.

Praktikum 14-3: Melacak nilai variabel dari halaman lain

- ☑ Jalankan aplikasi Text Editor
- ☑ Tuliskan kode berikut:

```

<?php
    session_start();
    session_register("nama");
    session_register("kelas");
    session_register("nilai");
?>

<html>
<head>
    <title>Menggunakan Session</title>
</head>
<body>
    <h2>Demo menggunakan session (Bagian 3)</h2>

    <?php
        $ _SESSION[ 'nama' ] = "Rizki Dwi Mulyana";
        $ _SESSION[ 'kelas' ] = "Web Programmer";
        $ _SESSION[ 'nilai' ] = array(
            "Algoritma" => 'A',
            "SQL" => 'A',
            "HTML & CSS" => 'B',
            "PHP" => 'A' );

        echo "Selamat Datang, " .

```

```

" <strong>" . $_SESSION['nama'] . "</strong>";
?>
<br /><br />
<a href="nilaikursus.php">Lihat nilai kursus</a>

</body>
</html>

```

- ☑ Simpan file dengan nama **session3.php**
- ☑ Buatlah file baru, dan tuliskan kode berikut:

```

<?php
    session_start();
?>
<html>
<head>
    <title>Daftar Nilai</title>
</head>
<body>
    <h3>Daftar Nilai Peserta Kursus</h3>
    Nama: <?php echo $_SESSION['nama']; ?><br />
    Kelas: <?php echo $_SESSION['kelas']; ?><br /><br />

    <?php
        $daftarnilai = $_SESSION['nilai'];
    ?>

    <table border="1">
        <tr>
            <th width="250">Materi Kursus</th>
            <th width="60">Nilai</th>
        </tr>
        <?php
            foreach($daftarnilai as $materi => $nilai) {
                ?>
                <tr>
                    <td><?php echo $materi; ?></td>
                    <td align="center"><?php echo $nilai; ?></td>
                </tr>
                <?php
            }
        ?>
    </table>

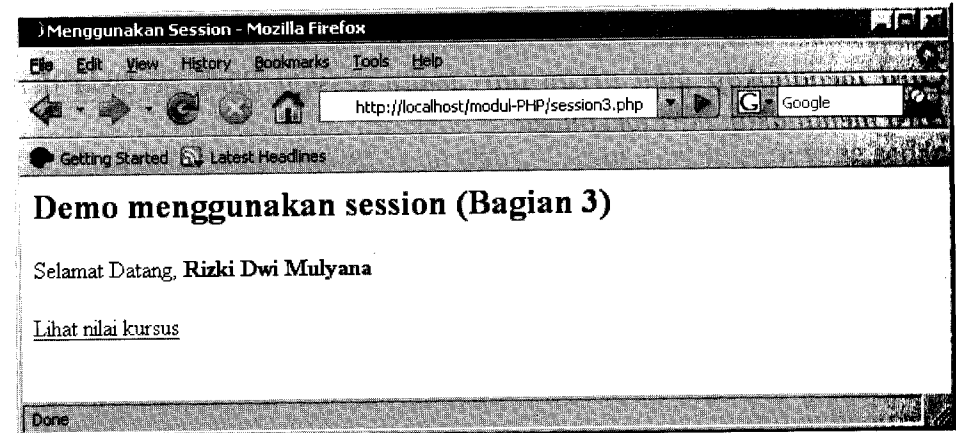
```

```

</body>
</html>

```

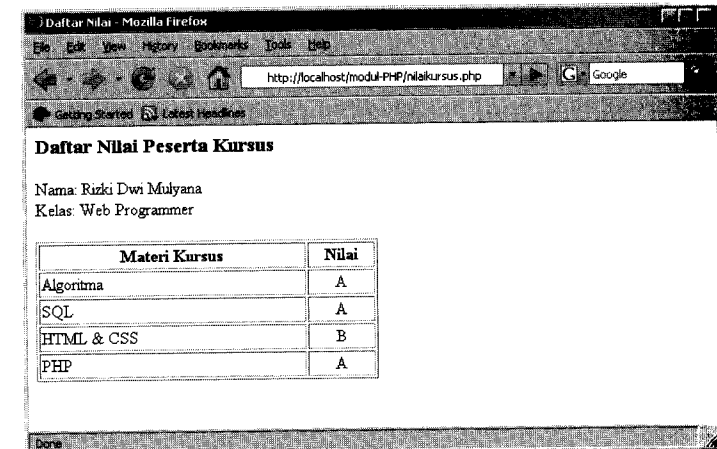
- ☑ Simpan file dengan nama **nilaikursus.php**
- ☑ Jalankan web browser dan masukkan **http://localhost/modul-PHP/session3.php**



- ☑ Klik link Lihat nilai kursus

Keterangan:

Pada praktikum kali ini kita mendaftarkan tiga buah variabel ke dalam session: nama, kelas, dan nilai. Variabel nama dan kelas bertipe string, sedangkan nilai bertipe array.




```
session_register("nama");
session_register("kelas");
session_register("nilai");
```

Seperti yang Anda lihat pada hasil di atas, nilai dari variabel-variabel yang kita daftarkan pada file **session3.php** ternyata masih dapat dikenali oleh file **nilaikursus.php**.

Cara menangkap nilai array dari suatu session tidak berbeda dengan cara menangkap variabel biasa:

```
$daftarnilai = $_SESSION['nilai'];
```

Setelah array di dalam session ditampung ke dalam variabel `$daftarnilai`, kita dapat dengan mudah mengekstrak nilainya dan memasukkannya ke dalam sebuah tabel. Dalam contoh ini, kita menggunakan statemen `foreach` untuk mengambil setiap nilai elemen array yang ada:

```
<table border="1">
  <tr>
    <th width="250">Materi Kursus</th>
    <th width="60">Nilai</th>
  </tr>
  <?php
  foreach($daftarnilai as $materi => $nilai) {
    ?>
    <tr>
      <td><?php echo $materi; ?></td>
      <td align="center"><?php echo $nilai; ?></td>
    </tr>
    <?php
  }
  ?>
</table>
```

Menghapus Variabel dari Suatu Session

Pada saat Anda tidak membutuhkan lagi variabel yang terdapat di dalam suatu session yang sedang aktif, Anda dapat menghapus variabel tersebut dengan cara memanggil fungsi `session_unregister()`. Bentuk umum fungsi ini adalah:

```
session_unregister(STRING)
```

STRING adalah nama variabel yang akan dihapus dari session yang sedang aktif.

Contoh:

```
if (session_is_registered("nama")) {
  unset("nama");
  session_unregister("nama");
}
```

`unset()` pada contoh di atas digunakan untuk menghapus nilai yang terdapat pada variabel `nama`.

Jika Anda ingin menghapus semua variabel yang telah terdaftar dalam suatu session, gunakan fungsi `session_unset()`. Bentuk umum fungsi ini adalah:

```
session_unset()
```

Berbeda dengan `session_unregister()`, fungsi `session_unset()` tidak memerlukan argumen pada saat pemanggilannya.

Mengakhiri Session

Anda dapat menggunakan fungsi `session_destroy()` untuk mengakhiri sebuah session, dengan menghapus semua variabel di dalam session tersebut. Fungsi ini tidak memiliki argumen. Bentuk umum fungsi ini adalah:

```
session_destroy()
```

Contoh penggunaan:

```
session_start();  
$_SESSION['test'] = 5;  
session_destroy();  
echo SESSION['test']; // mencetak nilai 5  
unset($_SESSION['test']);  
echo SESSION['test']; // nilai kosong
```

Bagian 3

Workshop

Modul
15

Workshop 1: Membuat Otentikasi User

Tinjauan Kasus

Jika Anda akan membangun suatu *website* yang di dalamnya terdapat proses registrasi *user*, maka Anda perlu menyediakan *form login* yang digunakan untuk otentikasi *user*. Dengan kata lain, hanya *user* yang sudah terdaftar saja yang dapat mengakses halaman-halaman di dalam *website* Anda.

Pada bagian ini, kita akan membuat suatu aplikasi sederhana untuk menangani proses otentikasi *user*. Di sini, *user* akan disimpan pada sebuah tabel di dalam *database*. Setiap *user* yang melakukan *login* melalui *form*

login akan diperiksa terlebih dahulu. Jika informasi yang dikirimkan sesuai dengan data di dalam *database*, maka *user* tersebut diizinkan untuk mengakses halaman berikutnya. Sebaliknya, jika informasi yang dikirimkan tidak ada di dalam *database* maka sistem akan menolak *user* tersebut dengan cara menampilkan kembali *form login* kepada *user* bersangkutan.

Dalam *workshop* ini, kita tidak akan menggunakan session di dalam proses otentikasi. Anda dapat menambahkan sendiri ke dalam aplikasi ini sesuai dengan kebutuhan yang Anda hadapi.

Membuat Tabel yang Diperlukan

Sebelum Anda mulai pembuatan aplikasi, terlebih dahulu Anda harus menyiapkan tabel *user* di dalam *database*. Anda dapat membuat *database dummy* untuk keperluan ini, misalnya dengan menggunakan kode berikut di dalam MySQL Client:

```
create database contohdb;
```

Gunakan perintah *USEDDB* untuk mengaktifkan *database* di atas:

```
use contohdb;
```

Kode untuk pembuatan tabel adalah:

```
create table webuser (  
  id varchar(15) not null,  
  nama varchar(25),  
  pwd varchar(40),  
  constraint pk_webuser primary key(id)  
);
```

Membuat Kode untuk Koneksi ke Database

Kode yang akan kita buat pada bagian ini berfungsi untuk melakukan koneksi ke *database* sehingga aplikasi PHP yang akan kita tulis nanti dapat berkomunikasi dengan tabel WEBUSER.

File ini akan kita gunakan pada setiap kode yang membutuhkan koneksi ke *database* dengan menggunakan fungsi `require()`.

Kode 15-1: connectdb.php

```
<?php  
  
$dbname = "contohdb";  
$host = "localhost"; //127.0.0.1  
$username = "bimo";  
$password = "p4NdaW4";  
  
$conn = mysqli_connect($host, $username, $password,  
                        $dbname);  
  
if (mysqli_connect_errno()) {  
  echo "Koneksi ke server database gagal.";  
  exit();  
}  
  
?>
```

Membuat Form Registrasi User

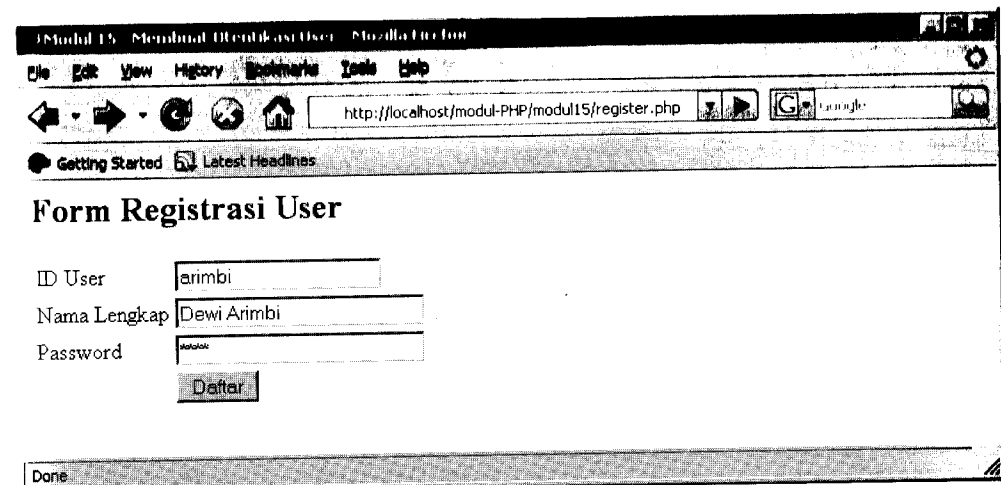
Form registrasi digunakan untuk melayani *user* baru yang ingin menjadi anggota di dalam *website* Anda. Informasi yang dikirimkan oleh *user* melalui *form* registrasi akan dicatat atau disimpan ke dalam tabel WEBUSER.

Kode yang diperlukan untuk membuat *form* registrasi adalah:

Kode 15-2: register.php

```
<html>
<head>
  <title>Modul 15 - Membuat Otentikasi User</title>
</head>
<body>
<h2>Form Registrasi User</h2>
<form action="do_register.php" method="post">
<table>
  <tr>
    <td>ID User</td>
    <td><input type="text" name="id"
      maxlength="15" size="15" /></td>
  </tr>
  <tr>
    <td>Nama Lengkap</td>
    <td><input type="text" name="nama"
      maxlength="25" size="25" /></td>
  </tr>
  <tr>
    <td>Password</td>
    <td><input type="password" name="pwd"
      maxlength="15" size="25" /></td>
  </tr>
  <tr>
    <td>&nbsp;</td>
    <td><input type="submit" name="daftar"
      value="Daftar" /></td>
  </tr>
</table>
</form>
</body>
</html>
```

Tampilan *form* registrasi:



Membuat Kode PHP untuk Menangani Proses Registrasi

Setelah membuat *form* registrasi, langkah selanjutnya adalah membuat kode PHP untuk menangani proses registrasi. Kode ini berfungsi untuk memasukkan informasi yang dimasukkan oleh *user* melalui *form* registrasi ke dalam tabel WEBUSER di dalam *database*.

Kode 15-3: do_register.php

```
<html>
<head>
  <title>Modul 15 - Membuat Otentikasi User</title>
</head>
<body>
<h2>Status Registrasi</h2>
<?php
if (isset($_POST['daftar'])) {
  $id   = $_POST['id'];
  $nama = $_POST['nama'];
  $pwd  = $_POST['pwd'];

  require("connectdb.php");
```

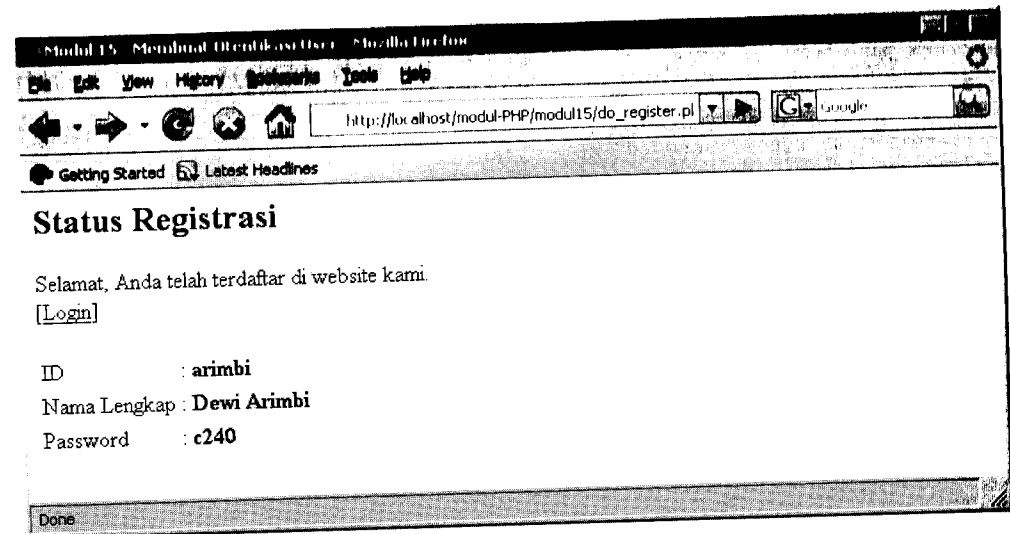


```

$sql = "insert into webuser(id, nama, pwd) ",
      "values('$id', '$nama', md5('$pwd'))";
mysqli_query($conn, $sql);
$num = mysqli_affected_rows($conn);
if ($num > 0) {
    ?>
    Selamat, Anda telah terdaftar di website kami.<br />
    [<a href="login.php">Login</a>]<br /><br />
    <table>
    <tr>
    <td>ID</td>
    <td>:</td>
    <td><strong><?php echo $id; ?></strong></td>
    </tr>
    <tr>
    <td>Nama Lengkap</td>
    <td>:</td>
    <td><strong><?php echo $nama; ?></strong></td>
    </tr>
    <tr>
    <td>Password</td>
    <td>:</td>
    <td><strong><?php echo $pwd; ?></strong></td>
    </tr>
    </table>
    <?php
} else {
    ?>
    Proses registrasi gagal. Silahkan ulangi!<br />
    [<a href="register.php">Kembali ke Form Register</a>]
    <?php
}
?>
</body>
</html>

```

Contoh tampilan pada saat program menangani proses registrasi:



Membuat Form Login

Form login adalah *form* yang digunakan oleh *user* untuk mengirimkan informasi ID dan *password*. Jika *user* salah memasukkan informasi yang diminta, maka sistem akan menolak *user* tersebut dengan menampilkan kembali *form login*.

Kode 15-2: login.php

```

<html>
<head>
    <title>Modul 15 - Membuat Otentikasi User</title>
</head>
<body>
    <h2>Form Login</h2>
    Anda belum terdaftar sebagai user?
    [<a href="register.php">Daftar sekarang</a>]
    <br /><br />
    <form action="do_login.php" method="post">
    <table>
    <tr>
    <td>ID User</td>
    <td><input type="textfield" name="id">

```



```

maxlength="15" size="25" /></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="pwd"
maxlength="15" size="25" /></td>
</tr>
<tr>
<td>&nbsp;</td>
<td><input type="submit" name="login"
value="Login" />
</td>
</tr>
</table>
</form>
</body>
</html>

```

Membuat Kode PHP untuk Menangani Proses Login

Setelah *form login* tersedia, langkah terakhir yang perlu dilakukan adalah membuat kode program untuk menangani proses *login* yang dilakukan oleh *user*.

Kode 15-3: do_login.php

```

<html>
<head>
<title>Modul 15 - Membuat Otentikasi User</title>
</head>
<body>
<?php
if (isset($_POST['login'])) {
    $id = $_POST['id'];
    $pwd = $_POST['pwd'];

    require("connectdb.php");

    $query = "select nama from webuser "
            . "where id = '$id' and pwd = md5('$pwd')";
    $result = mysqli_query($conn, $query);
    $num = mysqli_num_rows($result);

    if ($num == 1) {
        list($nama) = mysqli_fetch_array($result);
        ?>
        <h2>Secret Area</h2>
        Selamat datang,
        <strong><?php echo $nama; ?></strong><br />

        Ini adalah contoh halaman yang hanya dapat diakses oleh
        user yang telah terdaftar di dalam website.

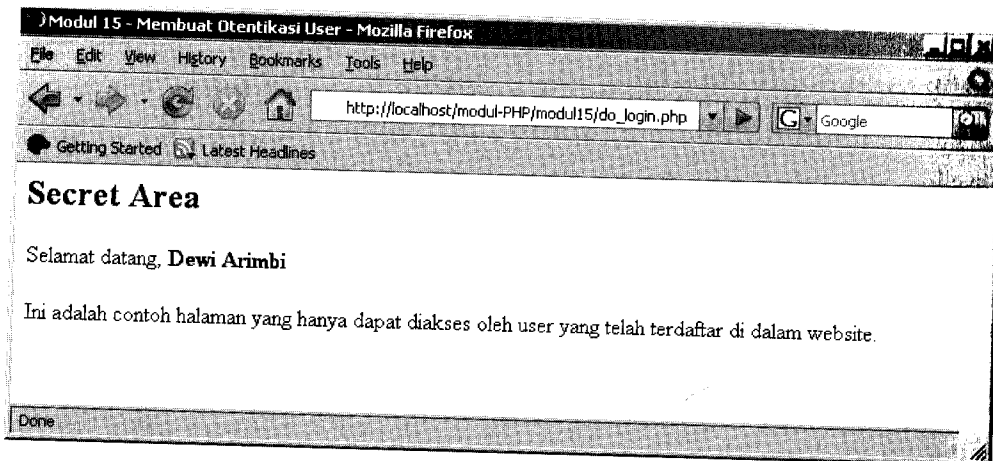
        <?php
    } else {
        header("Location: "
            . "http://localhost/modul-PHP/modul15/login.php");
    }
}
}

```

Tampilan *form login*:

```
?>
</body>
</html>
```

Jika *user* yang melakukan *login* telah terdaftar di dalam *database*, maka halaman yang akan ditampilkan adalah:



Cara Kerja Sistem

1. Mula-mula sistem akan menampilkan *form login* (file **login.php**) kepada *user*.
2. Bagi *user* yang belum terdaftar, mereka dapat mendaftarkan diri dengan memilih menu **Daftar sekarang**. Link tersebut akan memanggil *form* registrasi (file **register.php**). Ketika proses registrasi *user* berlangsung, sistem akan mengeksekusi file **do_register.php**.
3. Untuk melakukan *login* ke dalam *website*, *user* harus mengisikan informasi ID *User* dan *Password* terlebih dahulu melalui *form login*. Ketika tombol *Login* diklik, sistem akan mengeksekusi file **do_login.php**.

Workshop 2: Membuat Buku Tamu

Tinjauan Kasus

Buku tamu adalah salah satu fasilitas yang digunakan untuk menampung kesan dan pesan dari para pengunjung terhadap *website* yang kita kelola. Dari kritik dan saran para pengunjung tersebut kita dapat mengetahui kelebihan dan kekurangan *website* kita, apakah dari segi isi, tampilan, kelengkapan, maupun lainnya.

Dalam *workshop* ini, kita akan menyimpan data-data yang dimasukkan oleh *user* (para pengunjung) ke dalam sebuah tabel di dalam *database*.

Membuat Tabel yang Diperlukan

Untuk kebutuhan pembuatan buku tamu, kita hanya memerlukan *satu* tabel. Anda dapat membuat tabel tersebut melalui MySQL *Client* menggunakan kode berikut:

```
use contohdb;
create table bukutamu (
    id integer not null auto_increment,
    tanggal date,
    nama varchar(25),
    email varchar(50),
    url varchar(100),
    komentar text,
    constraint pk_bukutamu primary key(id)
);
```

Field ID digunakan sebagai pembeda antara data yang satu dengan yang lainnya. DATE digunakan untuk menentukan kapan *user* melakukan pengisian buku tamu. NAMA, EMAIL, dan URL digunakan untuk mencatat informasi *user*; sedangkan KOMENTAR digunakan untuk menyimpan komentar yang ditulis oleh *user*.

Membuat Kode untuk Koneksi ke Database

Isi dari kode ini sama seperti yang telah Anda tulis pada Modul 15, Anda dapat menyalin *file* tersebut dan menempatkannya ke dalam direktori `..\htdocs\modul-PHP\modul16`.

Kode 16-1: connectdb.php

```
<?php
$dbname = "contohdb";
$host = "localhost"; //127.0.0.1
$username = "bimo";
$password = "p4NdaW4";
```

```
$conn = mysqli_connect($host, $username, $password,
    $dbname);
```

```
if (mysqli_connect_errno()) {
    echo "Koneksi ke server database gagal.";
    exit();
}
```

```
?>
```

Membuat Menu Buku Tamu

Halaman awal buku tamu yang akan dibuat di sini terdiri dari dua buah menu:

- **Isi Buku Tamu**

Menu ini digunakan untuk menampilkan *form* entri buku tamu.

- **Lihat Daftar Komentar**

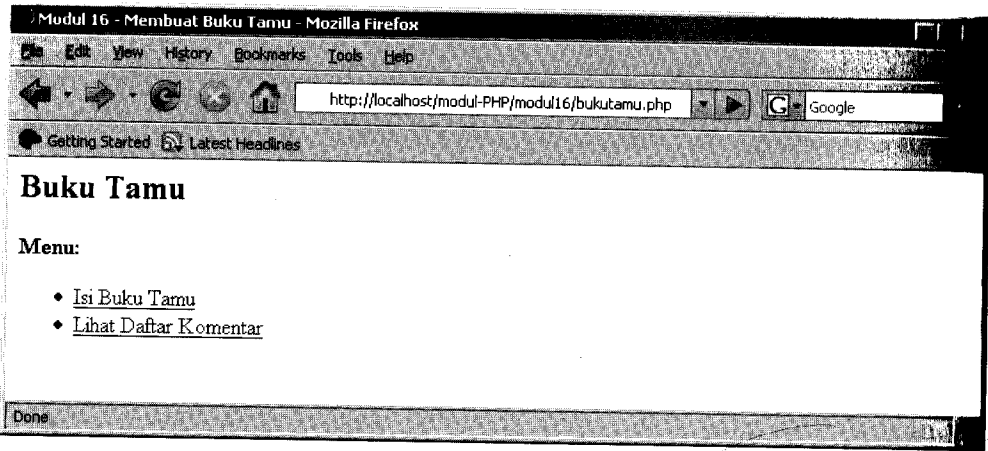
Menu ini digunakan untuk menampilkan daftar komentar yang telah disimpan di dalam buku tamu.

Kode 16-2: bukutamu.php

```
<html>
<head>
    <title>Modul 16 - Membuat Buku Tamu</title>
</head>
<body>
<h2>Buku Tamu</h2>
<strong>Menu:</strong>
<ul>
    <li><a href="addentry.php">Isi Buku Tamu</a></li>
    <li><a href="listentry.php">Lihat Daftar Komentar</a>
        </li>
</ul>
```

```
</body>
</html>
```

Tampilan menu buku tamu:



Membuat Form Entri

Form entri buku tamu dapat dibuat secara mudah, sama seperti yang sudah Anda lakukan pada praktikum-praktikum sebelumnya di dalam modul ini. Form entri yang akan kita buat hanya berisi informasi yang perlu user masukkan. *Field* ID dan TANGGAL akan dimasukkan secara otomatis melalui kode program.

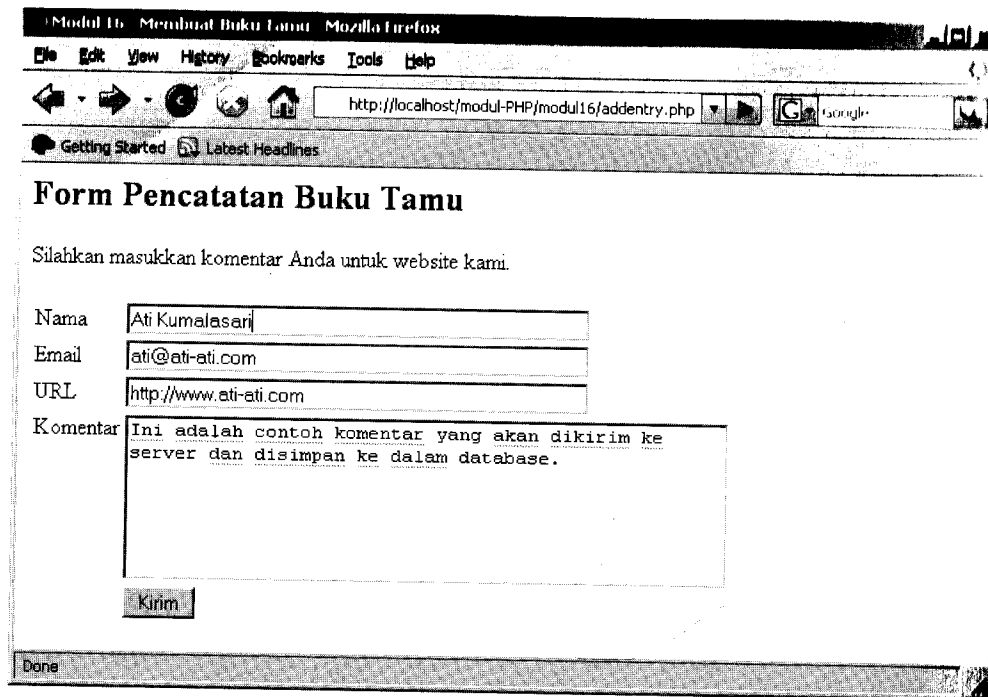
Berikut ini kode yang diperlukan untuk membuat *form* entri buku tamu:

Kode 16-3: addentry.php

```
<html>
<head>
  <title>Modul 16 - Membuat Buku Tamu</title>
</head>
<body>
<h2>Form Pencatatan Buku Tamu</h2>
```

```
Hilahkan masukkan komentar Anda untuk website kami.
<br /><br />
<form action="do_addentry.php" method="post">
<table>
  <tr>
    <td>Nama</td>
    <td><input type="textfield" name="nama"
      maxlength="25" size="50" /></td>
  </tr>
  <tr>
    <td>Email</td>
    <td><input type="textfield" name="email"
      maxlength="50" size="50" /></td>
  </tr>
  <tr>
    <td>URL</td>
    <td><input type="textfield" name="url"
      maxlength="100" size="50" /></td>
  </tr>
  <tr>
    <td valign="top">Komentar</td>
    <td>
      <textarea name="komentar" cols="50" rows="6">
      </textarea>
    </td>
  </tr>
  <tr>
    <td>&nbsp;</td>
    <td><input type="submit" name="kirim"
      value="Kirim" /></td>
  </tr>
</table>
</form>
</body>
</html>
```

Tampilan *form* entri yang dihasilkan oleh kode di atas:



```

<body>
<?php
if (isset($_POST['kirim'])) {
    $nama      = $_POST['nama'];
    $email     = $_POST['email'];
    $url       = $_POST['url'];
    $komentar  = $_POST['komentar'];

    require("connectdb.php");

    $tanggal = date('Y/m/d'); //Contoh: 2010/01/12

    $sql = "insert into bukutamu(tanggal, nama, email, ".
           "url, komentar) ".
           "values('$tanggal', '$nama', '$email', '$url', ".
           "'$komentar')";

    mysqli_query($conn, $sql);
    $num = mysqli_affected_rows($conn);

    if ($num > 0) {
        ?>
        <h2>Terima Kasih</h2>
        <strong><?php echo $nama; ?></strong>,
        komentar Anda telah terkirim ke server kami.<br />
        [<a href="listentry.php">Lihat Daftar Komentar</a>]
        <br /><br />
        <?php
    } else {
        ?>
        <h2>Error</h2>
        Proses pencatatan bukutamu gagal. Silahkan ulangi!
        <br />
        [<a href="addentry.php">
        Kembali ke Form Pencatatan Buku Tamu</a>]
        <?php
    }
}
?>
</body>
</html>

```

Membuat Kode PHP untuk Menangani Proses Entri

Kode yang akan dibuat pada bagian ini berfungsi untuk mengirim informasi yang *user* masukkan melalui *form* ke *server*, kemudian memasukkannya ke dalam tabel BUKUTAMU di dalam *database* CONTOHDB.

Kode yang diperlukan adalah sebagai berikut:

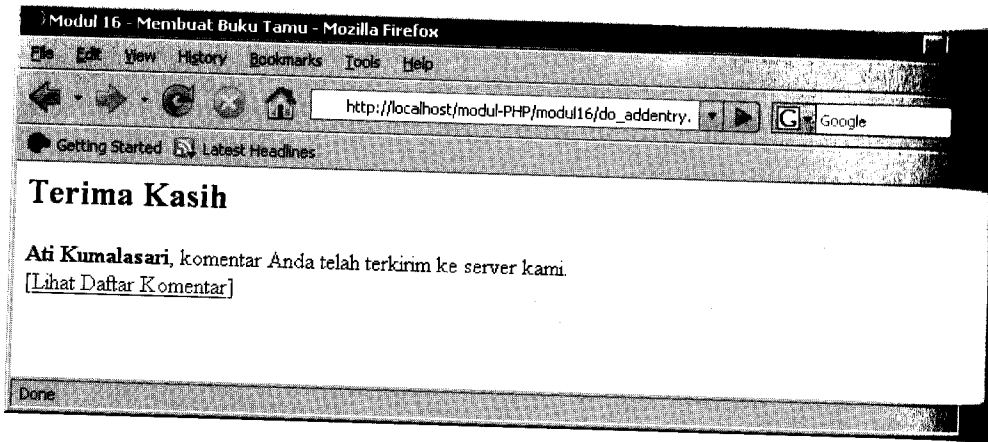
Kode 16-4: do_addentry.php

```

<html>
<head>
    <title>Modul 16 - Membuat Buku Tamu</title>
</head>

```


Contoh tampilan yang dihasilkan pada saat *user* mengirimkan komentar melalui *form* entri buku tamu:



Membuat Kode PHP untuk Menampilkan Komentar

Jika Anda akan mengizinkan *user* untuk melihat komentar yang dikirim oleh *user* lain, Anda dapat menampilkan daftar komentar tersebut ke dalam sebuah tabel.

Berikut ini kode untuk menampilkan komentar yang telah dimasukkan oleh *user*:

Kode 16-5: listentry.php

```
<html>
<head>
  <title>Modul 16 - Membuat Buku Tamu</title>
</head>
<body>
<h2>Daftar Komentar dalam Buku Tamu</h2>

<?php
```

```
require("connectdb.php");
```

```
$sql = "select tanggal, nama, email, url, komentar ".
      "from bukutamu order by tanggal desc";
```

```
$result = mysqli_query($conn, $sql);
$num = mysqli_num_rows($result);
```

```
if ($num > 0) {
  ?>
```

```
<table>
```

```
<tr bgColor="#ddcc45">
```

```
<th width="80">Tanggal</th>
```

```
<th width="140">Nama</th>
```

```
<th width="200">Email</th>
```

```
<th width="200">URL</th>
```

```
<th width="250">Komentar</th>
```

```
</tr>
```

```
<?php
```

```
while(list($tanggal, $nama, $email, $url, $komentar) =
      mysqli_fetch_array($result)) {
```

```
  ?>
```

```
<tr>
```

```
<td vAlign="top"><?php echo $tanggal; ?></td>
```

```
<td vAlign="top"><?php echo $nama; ?></td>
```

```
<td vAlign="top"><?php echo $email; ?></td>
```

```
<td vAlign="top">
```

```
<a href="<?php echo $url; ?>">
```

```
<?php echo $url; ?>
```

```
</a>
```

```
</td>
```

```
<td vAlign="top"><?php echo $komentar; ?></td>
```

```
</tr>
```

```
<?php
```

```
}
```

```
?>
```

```
</table>
```

```
<?php
```

```
) else {
```

```
  ?>
```

```
<i>Belum ada komentar.</i>
```

```
<?php
```

```
}
```

```
?>
```



```

<br /><br />
<div align="center">
[ <a href="addentry.php">Isi Buku Tamu</a>]
</div>

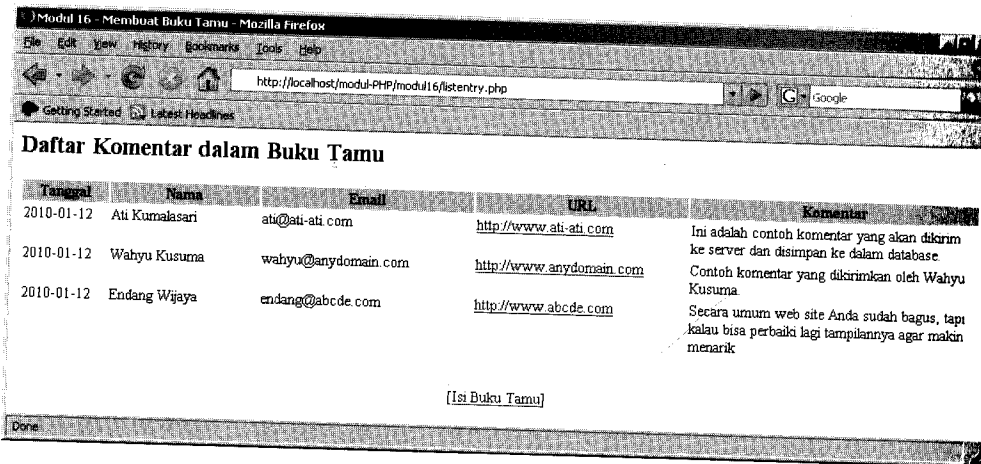
</body>
</html>

```



Referensi HTML

Contoh tampilan yang dihasilkan oleh kode di atas adalah:



Pengenalan HTML

Bagian ini akan menjelaskan dasar-dasar HTML, yang meliputi: pengenalan dokumen HTML, kerangka dokumen HTML, cara membuat dokumen HTML, dan cara menjalankan dokumen di dalam suatu *web browser*.

Dokumen HTML

HTML adalah singkatan dari **HyperText Markup Language**. HTML merupakan *file* teks yang ditulis menggunakan aturan-aturan kode tertentu untuk kemudian disajikan ke *user* melalui suatu aplikasi *web browser*. Setiap informasi yang tampil di *web* selalu dibuat menggunakan kode HTML. Oleh karena itu, dokumen HTML sering disebut juga sebagai *web page* (halaman *web*). Untuk membuat dokumen HTML, kita tidak tergantung pada aplikasi tertentu; karena dokumen HTML dapat dibuat menggunakan aplikasi Text Editor apa pun, bisa Notepad (untuk lingkungan MS Windows), Emacs atau Vi Editor (untuk lingkungan Linux), dan sebagainya.

Agar lebih mempermudah dalam pembuatan dokumen HTML, sekarang telah banyak tersedia aplikasi HTML Editor, yaitu suatu aplikasi Text Editor yang dikhususkan untuk pembuatan kode-kode HTML.

Kode HTML dari suatu halaman *web* dapat kita lihat dari aplikasi *web browser* yang kita gunakan, yaitu dengan cara memilih menu **View-Source** (untuk Internet Explorer) dan **View-Page Source** (untuk Mozilla Firefox).

Penamaan Dokumen HTML

Tidak ada aturan yang baku untuk penamaan suatu dokumen HTML. Artinya, kita bebas memilih nama untuk dokumen tersebut. Hanya ekstensinya saja yang perlu untuk diperhatikan. Suatu *file* teks akan dianggap sebagai dokumen HTML jika memiliki ekstensi **“.htm”** atau **“.html”**. Maka dari itu, setiap membuat dokumen HTML, kita harus menyimpannya ke dalam salah satu ekstensi tersebut, misalnya **Katalog.htm**, **Produk.htm**, **Servis.html**, **Kontak.html**, dan lain sebagainya.

Kerangka Dokumen HTML

Secara umum, dokumen HTML terbagi atas dua bagian, yaitu bagian **header** (kepala) dan **body** (badan). Bagian *header* diawali dengan *tag* `<head>` dan diakhiri dengan *tag* `</head>`, sedangkan bagian *body* diawali dengan *tag* `<body>` dan ditutup dengan *tag* `</body>`. Kedua bagian tersebut diapit oleh *tag* `<html>` dan `</html>`, yang digunakan untuk menandai bahwa dokumen yang kita buat adalah dokumen HTML. Untuk lebih jelasnya, berikut ini kerangka dari dokumen HTML.

```
<html> <!-- Tag untuk memulai dokumen HTML -->
```

```
<!-- Bagian header -->
```

```
<head>
```

```
<title>Judul halaman</title>
```

```
</head>
```

```
<!-- Bagian body -->
```

```
<body>
```

Informasi yang akan ditampilkan pada *web browser* harus ditulis dalam bagian ini.

```
</body>
```

```
</html> <!-- Tag untuk mengakhiri dokumen HTML -->
```

Seperti yang tampak di atas, *tag-tag* dalam dokumen HTML dibuat secara berpasangan (meskipun tidak semua). Sebagai contoh, dokumen HTML diawali dengan *tag* `<html>` dan ditutup atau diakhiri dengan *tag* `</html>`. Begitu juga dengan *tag* `<head>`, yang juga ditutup menggunakan *tag* `</head>`; serta *tag* `<body>` yang ditutup menggunakan *tag* `</body>`.

Bagian teks yang diapit oleh tanda `<!--` dan `-->` merupakan komentar, yaitu bagian teks yang tidak ikut ditampilkan ke dalam *web browser*.

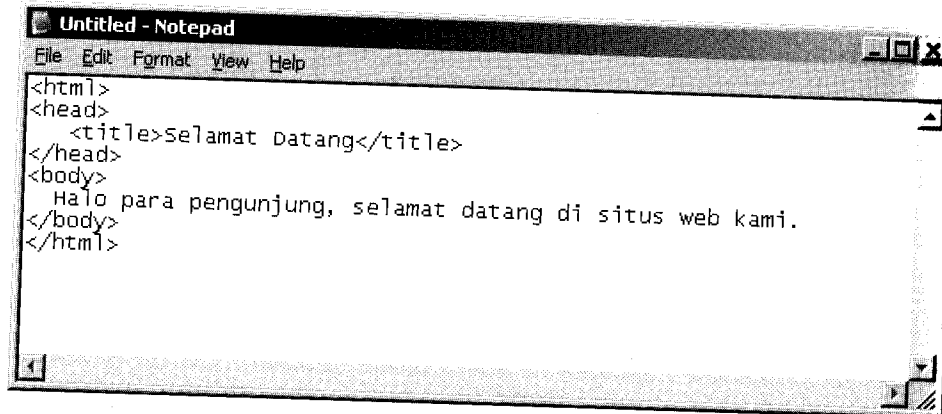
Membuat Dokumen HTML

Agar Anda bisa lebih jelas dalam memahami kerangka dokumen di atas, pada bagian ini kita akan mencoba untuk membuat dokumen HTML sederhana. Silakan ikuti langkah-langkah berikut ini.

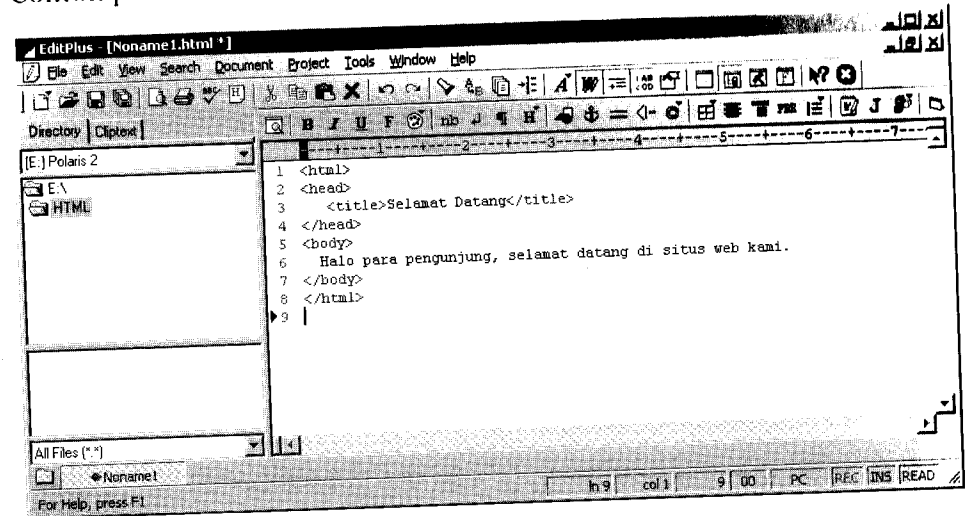
1. Jalankan suatu aplikasi Text Editor, misalnya Notepad, SCite, atau yang lainnya.
2. Tuliskan kode berikut ke dalam Text Editor bersangkutan.

```
<html>
<head>
  <title>Selamat Datang</title>
</head>
<body>
  Halo para pengunjung, selamat datang di situs web kami.
</body>
</html>
```

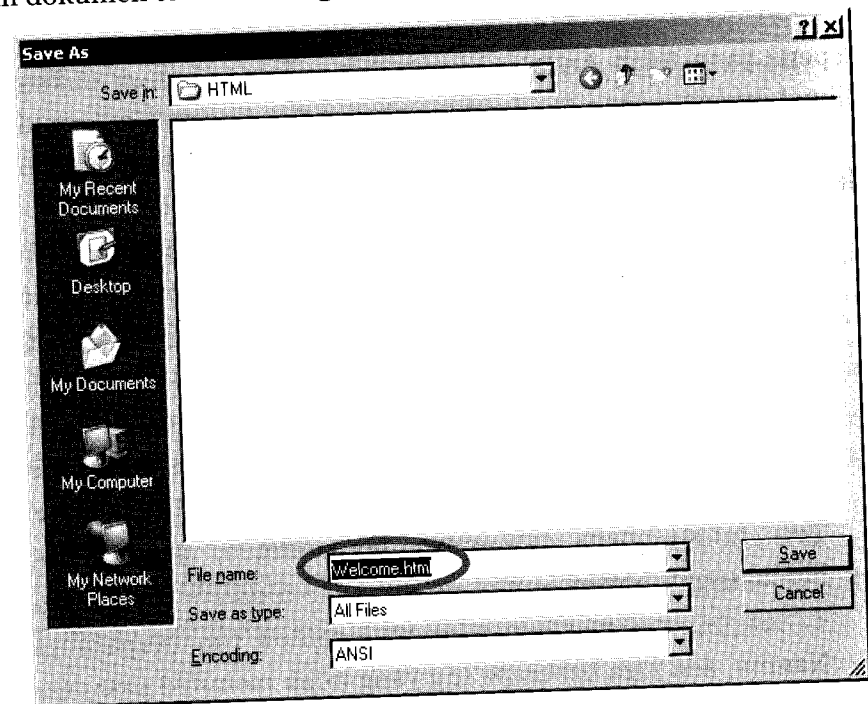
Contoh penulisan kode menggunakan Notepad:



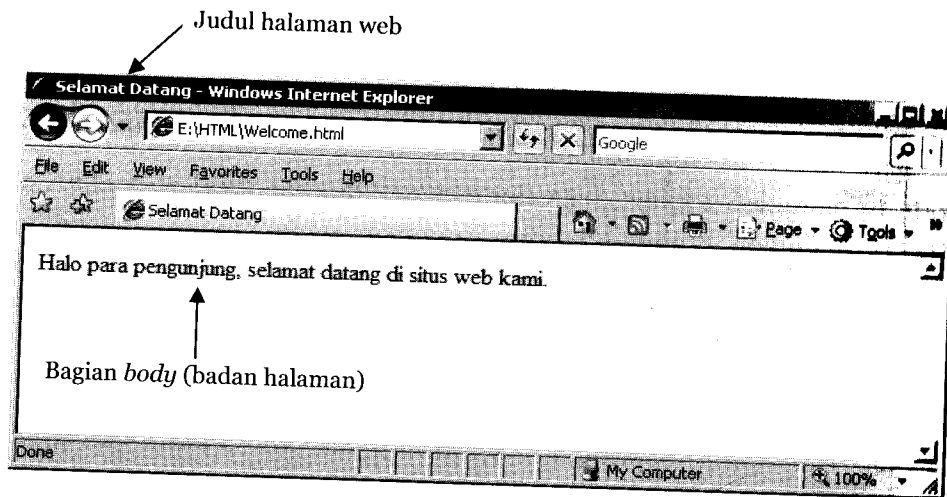
Contoh penulisan kode menggunakan salah satu aplikasi HTML Editor:



3. Simpan dokumen tersebut dengan nama **Welcome.html**.



4. Buka file Welcome.html melalui web browser.



Sekilas tentang Tag HTML

Seperti yang telah Anda pelajari sekilas di atas bahwa dokumen HTML dibuat menggunakan *tag-tag* khusus, dan dengan aturan yang khusus pula. Terdapat beberapa hal yang perlu diperhatikan dalam penggunaan *tag* di dalam dokumen HTML, yaitu:

- Tag HTML selalu diapit oleh tanda kurung siku, < dan >
- Meskipun kode HTML **tidak** bersifat *case-sensitive* (membedakan penggunaan huruf besar dan kecil), namun tag HTML sebaiknya ditulis menggunakan huruf kecil, misalnya <html>, <head>, </head>, <title>, dan sebagainya. Ini merupakan cara yang direkomendasikan oleh W3C, meskipun sebenarnya bisa saja ditulis menggunakan huruf besar, seperti: <HTML>, <HEAD>, </HEAD>, <TITLE>, dan sebagainya.
- Tag HTML pada umumnya berpasangan, seperti: tag <html> berpasangan dengan tag </html>, <head> berpasangan dengan </head>, dan masih banyak lagi contoh tag berpasangan lainnya. Namun, ada juga beberapa tag yang tidak memiliki pasangan, misalnya

tag
, <hr>, <input>, <meta>, dan lain-lain. Untuk menandai bahwa tag tersebut tidak memiliki pasangan, maka sebaiknya kita membubuhkan tanda *slash* (garis miring) di belakang tag tersebut, misalnya
, <hr />.

- Untuk penggunaan tag bersarang, maka penulisannya harus terurut. Sebagai contoh, perhatikan kode berikut:

```
<b>Anda harus <i>belajar</i> dengan giat.</b>
```

Pada contoh di atas, di dalam tag terdapat penggunaan tag lain lagi, yaitu tag <i>. Dalam kasus seperti ini, tag <i> harus ditutup (menggunakan tag </i>) terlebih dahulu sebelum kita menutup tag (menggunakan tag).

Atribut Tag

Pada umumnya, tag-tag dalam HTML memiliki beberapa atribut yang nilainya bisa ditentukan sesuai keperluan. Atribut suatu tag harus ditulis di dalam tag bersangkutan (sebelum tanda >). Berikut ini contoh tag dengan beberapa atribut di dalamnya.

```
<body bgColor="#ffffee" alink="#cccccc">
```

Pada contoh kode di atas, kita menentukan atribut `bgColor` dan `alink` dari tag <body>. Atribut `bgColor` digunakan untuk menentukan warna layar (*background*) dari suatu halaman *web* dan `alink` digunakan untuk menentukan warna *link* yang aktif dalam suatu halaman *web*, sedangkan `#ffffee` dan `#cccccc` merupakan nilai dari atribut tersebut.

Dari contoh yang ditampilkan di atas, kita bisa simpulkan bahwa bentuk umum penulisan atribut dari suatu tag adalah sebagai berikut:

```
<namaTag atribut1="NilaiAtribut1" atribut2 "NilaiAtribut2" ... >
```

Berikut ini contoh kode yang akan menunjukkan penggunaan atribut `bgColor` di dalam tag `<body>`.

```
<html>
<head>
  <title>Atribut Tag</title>
</head>
<body bgColor="#ffffee">
  Demo menggunakan atribut tag
</body>
</html>
```

Tag Umum dalam HTML

Setelah memahami secara umum tentang pembuatan dokumen HTML, pada bagian ini kita akan melanjutkan pembahasan tentang penggunaan *tag-tag* umum (*tag-tag* yang sering digunakan) secara lebih detail.

➤ Tag `<html>`

Seperti yang telah dikemukakan sebelumnya, dokumen HTML selalu diawali dengan tag `<html>` dan diakhiri dengan tag `</html>`. Tag inilah berperan sebagai penanda bahwa suatu dokumen dikatakan sebagai dokumen HTML.

➤ Tag `<head>`

Tag *head* berfungsi untuk menentukan informasi-informasi di bagian *header* (kepala) dokumen. Satu yang bersifat umum adalah judul dokumen, yang dibuat menggunakan tag `<title>`, yang berpasangan dengan tag

`</title>`. Karena judul berada dalam bagian *header*, maka penulisan tag `<title>` pun harus berada di dalam tag `<head>`, seperti berikut:

```
<head>
  <title>Contoh judul</title>
</head>
```

Tag penutup untuk *header*, `</head>`, harus ditulis sebelum tag `<body>`.

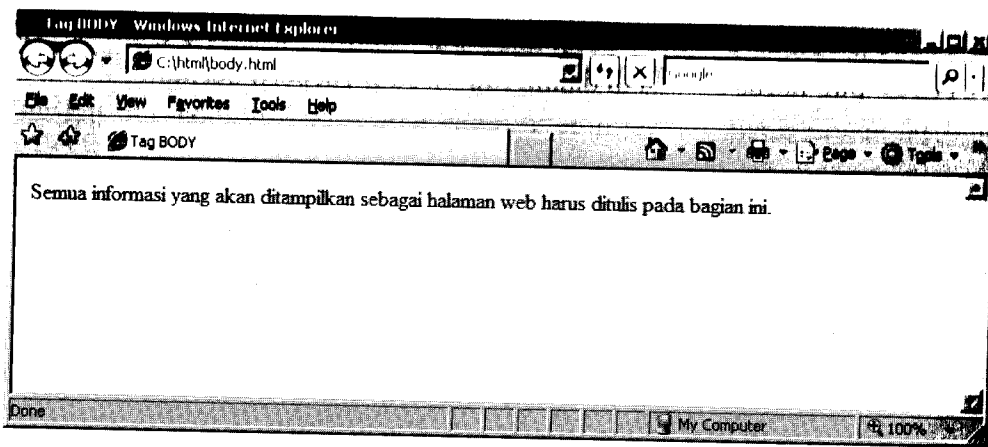
```
<head>
  <title>Contoh judul</title>
</head>
<body>
  ...
</body>
```

Bagian *header* dalam dokumen HTML tidak hanya terdiri atas judul saja. Terdapat beberapa tag lain yang dapat dijadikan *header* seperti `<meta>`, `<link>`, dan lain-lain.

➤ Tag `<body>`

Tag `<body>` merupakan awal bagian badan dokumen. Artinya, semua informasi yang ditulis setelah tag ini akan tampil sebagai halaman *web* (akan ditampilkan di layar *web browser*). Setelah selesai menuliskan bagian badan dokumen, *programmer* harus menutupnya dengan tag `</body>`.

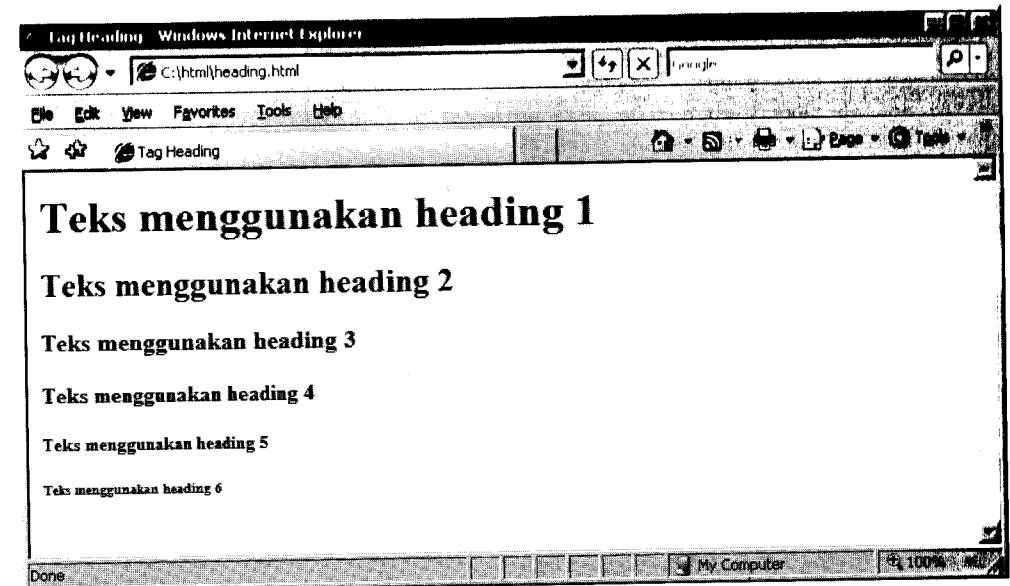
```
<html>
<head>
  <title>Tag BODY</title>
</head>
<body>
  Semua informasi yang akan ditampilkan sebagai halaman web harus ditulis pada
  bagian ini.
</body>
</html>
```



Membuat Heading (Judul Teks)

HTML menyediakan enam jenis judul teks. Jenis-jenis tersebut dibedakan berdasarkan ukuran hurufnya. *Tag* yang digunakan untuk membuat judul adalah `<hn>`, berpasangan dengan `</hn>`, di mana *n* adalah bilangan 1 sampai 6, yang merupakan jenis dari judul bersangkutan. Jenis pertama, `<h1>`, merupakan jenis judul yang memiliki ukuran huruf paling besar, disusul dengan jenis kedua, ketiga, dan seterusnya. Berikut ini contoh kode yang akan menunjukkan perbedaan jenis judul yang dimaksud.

```
<html>
<head>
  <title>Tag Heading</title>
</head>
<body>
<h1>Teks menggunakan heading 1</h1>
<h2>Teks menggunakan heading 2</h2>
<h3>Teks menggunakan heading 3</h3>
<h4>Teks menggunakan heading 4</h4>
<h5>Teks menggunakan heading 5</h5>
<h6>Teks menggunakan heading 6</h6>
</body>
</html>
```



Membuat Paragraf dalam Dokumen HTML

Paragraf dalam dokumen HTML dibuat menggunakan pasangan *tag* `<p>` dan `</p>`. Dalam dokumen HTML, sebuah paragraf ditandai dengan baris baru dengan jarak satu spasi. Berikut ini potongan kode untuk membuat paragraf dalam HTML.

```
<p>Ini adalah teks yang berada pada paragraf pertama</p>
<p>Ini adalah teks yang berada pada paragraf kedua</p>
<p>Ini adalah teks yang berada pada paragraf ketiga</p>
```

Untuk membuktikannya, coba Anda perhatikan contoh kode HTML di bawah ini.

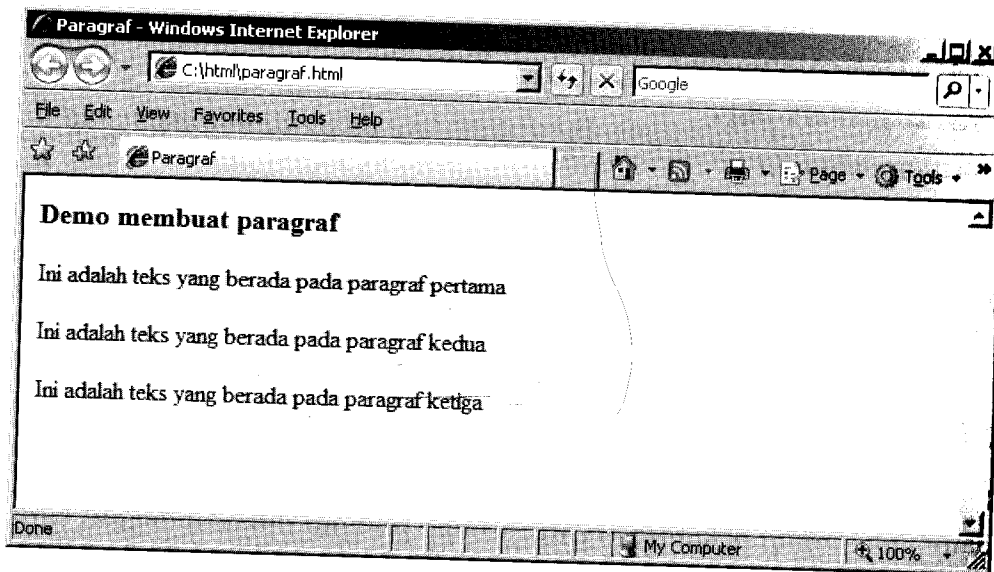
```
<html>
<head>
  <title>Paragraf</title>
</head>
<body>
```



```
<h3>Demo membuat paragraf</h3>
```

```
<p>Ini adalah teks yang berada pada paragraf pertama</p>
<p>Ini adalah teks yang berada pada paragraf kedua</p>
<p>Ini adalah teks yang berada pada paragraf ketiga</p>
```

```
</body>
</html>
```



Nilai Atribut ALIGN

Keterangan

Justify

Menentukan teks rata kiri-kanan

Center

Menentukan teks agar berada di tengah-tengah layar (secara horizontal)

Atribut ALIGN dapat disisipkan ke dalam ke *tag-tag* yang akan diatur posisi teksnya. Sebagai contoh, ALIGN dapat disisipkan ke dalam *tag* `<h>`, yang berarti bahwa kita akan mengatur posisi dari suatu judul. Begitu juga jika ingin mengatur posisi teks dalam suatu paragraf maka kita perlu menyisipkan atribut ALIGN ke dalam *tag* `<p>`. Berikut ini contoh kode yang akan menunjukkan cara mengatur posisi teks dalam dokumen HTML.

```
<html>
<head>
  <title>Text Alignment</title>
</head>
<body>
<h3 align="center">Demo "Text Alignment"</h3>
<p>Ini adalah teks yang secara default akan dianggap sebagai rata kiri.
Mengapa demikian? Karena HTML akan menganggap nilai atribut ALIGN yang tidak
dituliskan secara eksplisit dengan nilai LEFT.</p>

<p align="right">Ini adalah teks yang diatur dengan format rata kanan. Teks
seperti ini jarang digunakan atau jarang dijumpai dalam dokumen-dokumen
HTML.</p>

<p align="justify">Ini adalah teks yang diatur dengan format rata-kiri
kanan, yaitu dengan memasukkan nilai JUSTIFY pada atribut ALIGN. Teks dengan
format seperti ini cocok sekali digunakan untuk dokumen HTML yang sifatnya
formal.</p>

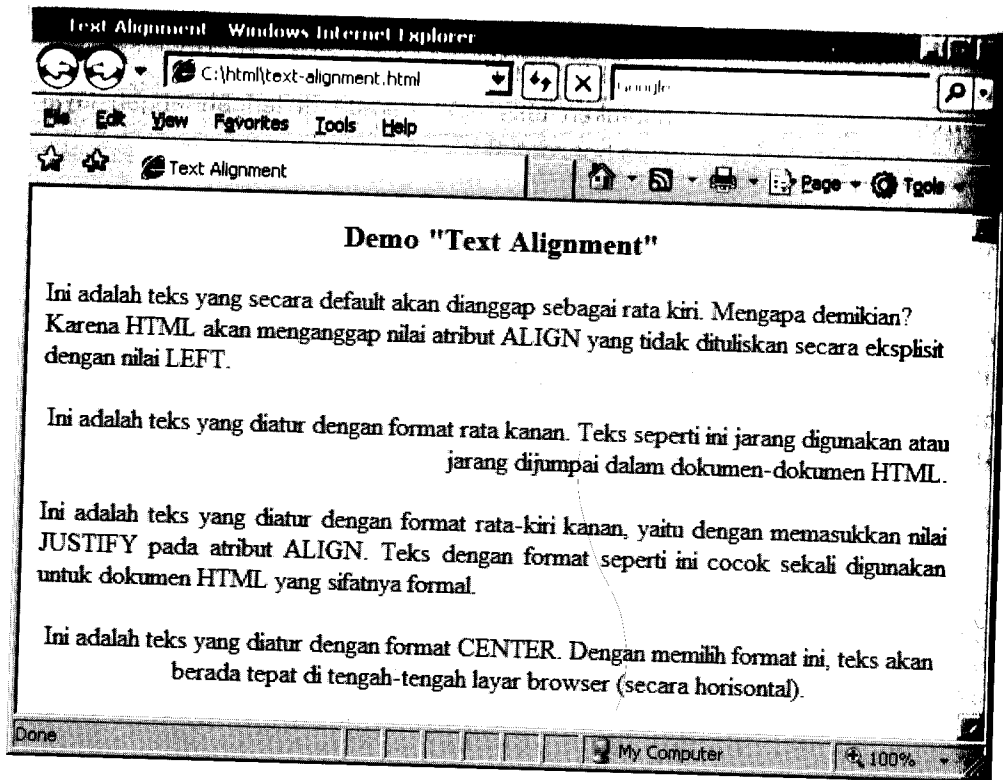
<p align="center">Ini adalah teks yang diatur dengan format CENTER. Dengan
memilih format ini, teks akan berada tepat di tengah-tengah layar browser
(secara horizontal).</p>

</body>
</html>
```

Text Alignment

Dalam membuat dokumen HTML, kita dapat menentukan letak dari teks atau gambar yang terdapat pada halaman *web*; apakah ingin rata kiri, rata kanan, rata kiri-kanan ataukah berada di tengah-tengah layar. Untuk melakukan hal ini, kita dapat mengisi atribut ALIGN dengan nilai-nilai berikut:

Nilai Atribut ALIGN	Keterangan
Left	Menentukan teks rata kiri
Right	Menentukan teks rata kanan



Membuat Baris Baru

Baris baru yang ditulis di dalam kode HTML tidak akan berpengaruh terhadap hasil yang akan ditampilkan di layar *web browser*. Sebagai contoh, perhatikan terlebih dahulu kode HTML berikut.

```
<html>
<head>
  <title>Baris Baru</title>
</head>
<body>
Baris pertama
Baris kedua
Baris ketiga
Baris keempat
```

```
Baris kelima
</body>
</html>
```

Pada kode di atas, teks “Baris pertama” sampai “Baris kelima” ditulis pada baris yang berbeda. Namun kenyataannya, hasil yang diperoleh dalam *web browser* adalah sebagai berikut.

Jadi, dapat kita simpulkan bahwa penulisan baris baru di dalam kode tidak ada pengaruhnya terhadap tampilan dokumen. Oleh karena itu, HTML menyediakan *tag* khusus untuk melakukan pembuatan baris baru di dalam dokumen, yaitu `<br>`. *Tag* ini tidak memiliki pasangan, sehingga kita perlu menyisipkan tanda garis miring (*slash*) di dalam *tag* tersebut, yang menandakan bahwa *tag* tersebut tidak memiliki pasangan. Bentuk penulisan *tag* `<br>` yang benar adalah:

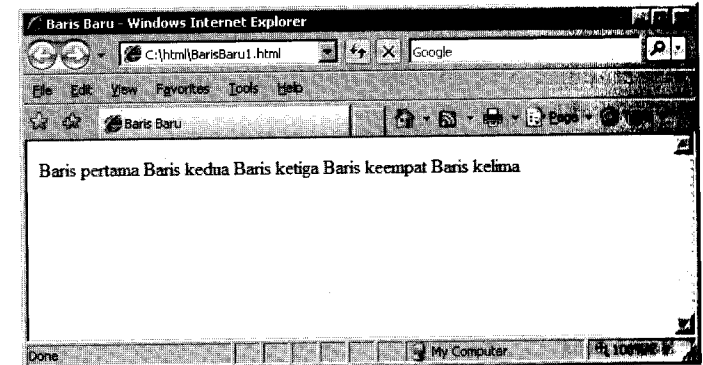
```
<br />
```

Sekarang, marilah kita lihat pengaruh *tag* tersebut terhadap kode HTML yang kita buat.

```
<html>
<head>
  <title>Baris Baru</title>
</head>
<body>

<n3>Demo membuat baris baru</h3>

Baris pertama<br />
```

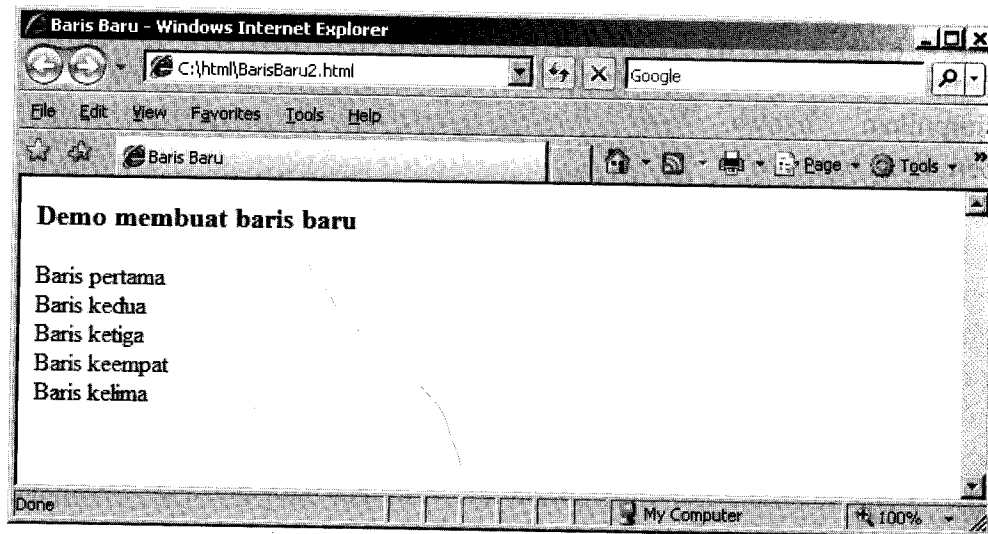


```

Baris kedua<br />
Baris ketiga<br />
Baris keempat<br />
Baris kelima

</body>
</html>

```



Dari hasil di atas tampak jelas bahwa fungsi *tag* `<br>` dalam kode HTML itu sebenarnya sama seperti fungsi tombol **Enter** (*line break*) ketika kita sedang bekerja pada suatu aplikasi Text Editor atau *Word Processor*. Sekarang coba Anda amati hasil yang akan diperoleh jika kode di atas ditulis seperti berikut:

```

<html>
<head>
  <title>Baris Baru</title>
</head>
<body>

<h3>Demo membuat baris baru</h3>

```

```

Baris pertama<br />Baris kedua<br />Baris ketiga<br />Baris keempat<br />Baris kelima

</body>
</html>

```

Menentukan Format Teks

Format teks yang terdapat pada dokumen *web* dapat kita atur sendiri formatnya sesuai dengan keperluan. Format yang umum yang sering dijumpai adalah penebalan teks (*bold*), teks miring (*italic*), teks bergaris bawah (*underline*), dan sejenisnya.

HTML menyediakan *tag-tag* khusus untuk menangani kebutuhan semacam ini. *Tag-tag* ini pada umumnya merupakan *tag* yang berpasangan. Berikut ini pembahasan tentang masing-masing format yang telah disebutkan di atas.

Membuat Teks Tebal

Untuk menebalkan teks atau karakter dalam suatu dokumen HTML, kita dapat menggunakan *tag* `<b>`, yang berpasangan dengan `</b>`. Setiap teks yang diapit oleh *tag-tag* tersebut akan ditampilkan sebagai teks dengan format tebal. Berikut ini contoh kode yang akan menunjukkan proses penebalan teks dalam dokumen.

```

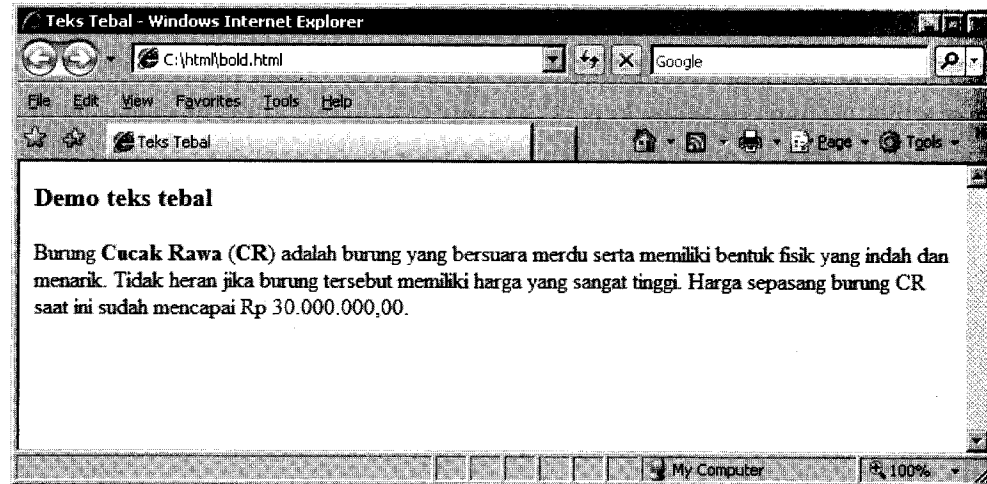
<html>
<head>
  <title>Teks Tebal</title>
</head>
<body>

<h3>Demo teks tebal</h3>

```

Burung Cucak Rawa (CR) adalah burung yang bersuara merdu serta memiliki bentuk fisik yang indah dan menarik. Tidak heran jika burung tersebut memiliki harga yang sangat tinggi. Harga sepasang burung CR saat ini sudah mencapai Rp 30.000.000,00.

```
</body>
</html>
```



Seperti yang tampak pada kode dan hasil di atas, kita dapat memilih mana teks yang akan ditebalkan dengan cara mengapit bagian teks tersebut dengan tag dan .

Membuat Teks Miring

Teks atau karakter dengan format miring dalam dokumen HTML dibuat menggunakan tag <i>, yang berpasangan dengan tag </i>. Berikut ini hasil modifikasi dari contoh kode di atas yang akan menunjukkan cara membuat teks miring.

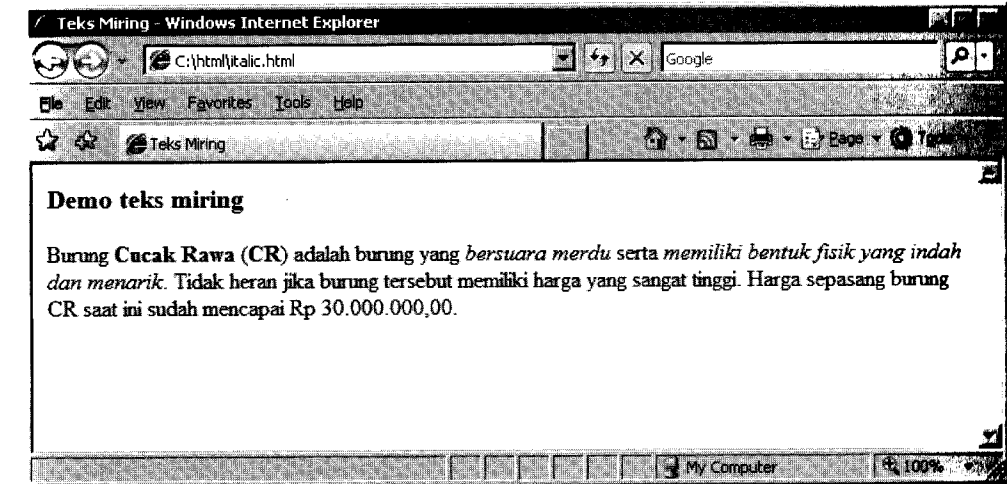
```
<html>
<head>
  <title>Teks Miring</title>
</head>
```

```
<body>
```

```
<h3>Demo teks miring</h3>
```

Burung Cucak Rawa (CR) adalah burung yang <i>bersuara merdu</i> serta <i>memiliki bentuk fisik yang indah dan menarik</i>. Tidak heran jika burung tersebut memiliki harga yang sangat tinggi. Harga sepasang burung CR saat ini sudah mencapai Rp 30.000.000,00.

```
</body>
</html>
```



Membuat Teks Bergaris Bawah

Untuk menjadikan suatu teks dalam dokumen HTML menjadi bergaris bawah, kita perlu menggunakan tag <u>, yang berpasangan dengan </u>. Teks ini pada umumnya digunakan untuk penegasan tertentu, misalnya seperti yang tampak pada contoh berikut.

```
<html>
<head>
  <title>Menebalkan Teks</title>
</head>
```

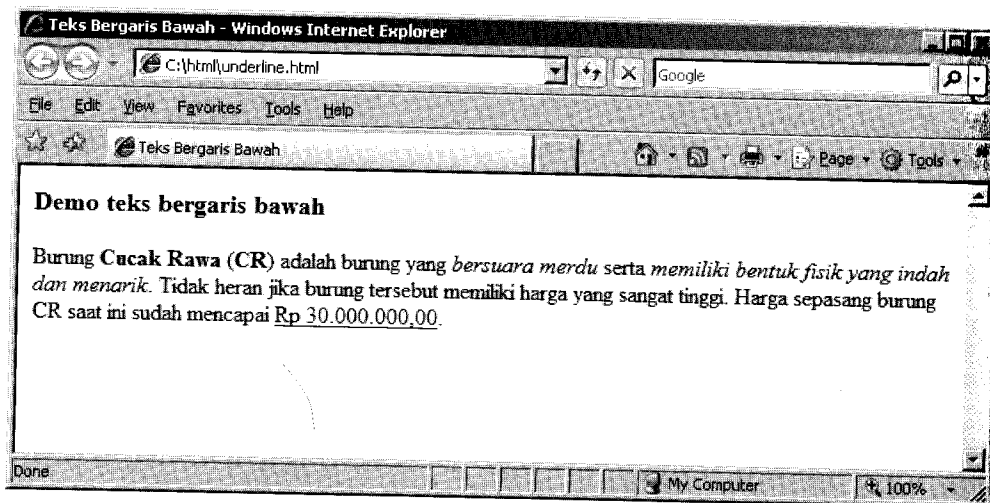
```
<body>
```

```
<h3>Demo menebalkan teks</h3>
```

```
Burung <b>Cucak Rawa</b> (<b>CR</b>) adalah burung yang <i>bersuara  
merdu</i> serta <i>memiliki bentuk fisik yang indah dan menarik</i>. Tidak  
heran jika burung tersebut memiliki harga yang sangat tinggi. Harga sepasang  
burung CR saat ini sudah mencapai <u>Rp 30.000.000,00</u>.
```

```
</body>
```

```
</html>
```



Mengombinasikan Format Teks

Masing-masing format teks di atas dapat dikombinasikan sehingga suatu teks tertentu bisa saja memiliki beberapa format sekaligus. Hal yang perlu diingat untuk melakukan hal ini adalah masalah urutan penempatan *tag*. Penempatan *tag* harus sesuai dengan urutannya. Perhatikan contoh potongan kode berikut:

```
<b><i>Teks</i></b>
```

Kode di atas digunakan untuk membuat teks menjadi tebal dan dicetak miring. Ini merupakan kombinasi dari dua format. Perhatikan urutannya, tidak boleh ditulis seperti berikut:

```
<b><i>Teks</b></i>
```

Berikut ini contoh kode yang menunjukkan cara mengombinasikan format teks dalam dokumen HTML.

```
<html>
```

```
<head>
```

```
<title>Kombinasi Format Teks</title>
```

```
</head>
```

```
<body>
```

```
<h3>Demo mengkombinasikan format teks</h3>
```

```
<p>Teks normal (tidak memiliki format apapun)</p>
```

```
<p><b>Teks tebal</b></p>
```

```
<p><b><i>Teks tebal dan miring</i></b></p>
```

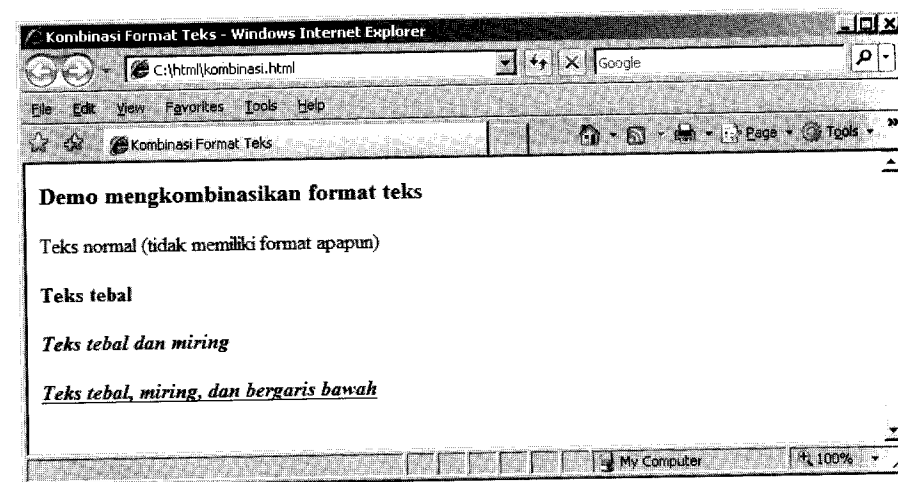
```
<p><b><i><u>
```

```
Teks tebal, miring, dan bergaris bawah
```

```
</u></i></b></p>
```

```
</body>
```

```
</html>
```



Membuat Format Superscript

Format *superscript* adalah format teks yang diperkecil ukuran hurufnya dan posisinya dinaikkan ke atas, seperti pada angka yang digunakan untuk perpangkatan. Sebagai contoh, 5^2 . Teks tersebut adalah lima kuadrat atau 5 pangkat 2. Angka 2 pada teks tersebut adalah teks dengan format *superscript*.

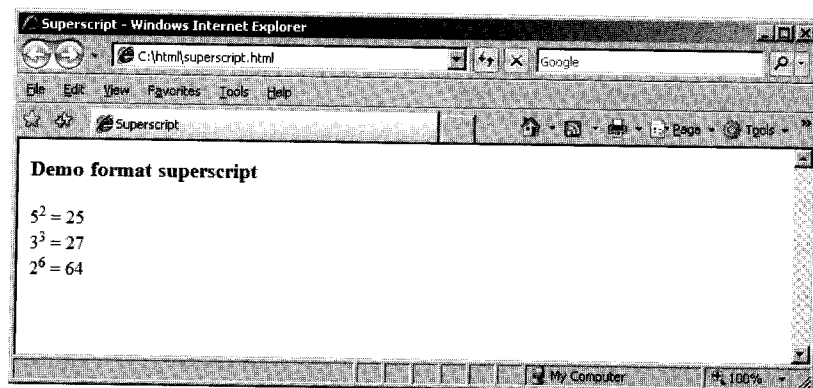
Untuk keperluan semacam ini, HTML menyediakan tag `<sup>`, yang berpasangan dengan `</sup>`. Setiap teks atau huruf yang diapit oleh tag-tag tersebut akan memiliki format *superscript*. Berikut ini contoh kode yang menunjukkan hal tersebut.

```
<html>
<head>
  <title>Superscript</title>
</head>
<body>

<h3>Demo format superscript</h3>

5<sup>2</sup> = 25<br />
3<sup>3</sup> = 27<br />
2<sup>6</sup> = 64<br />

</body>
</html>
```



Membuat Format Subscript

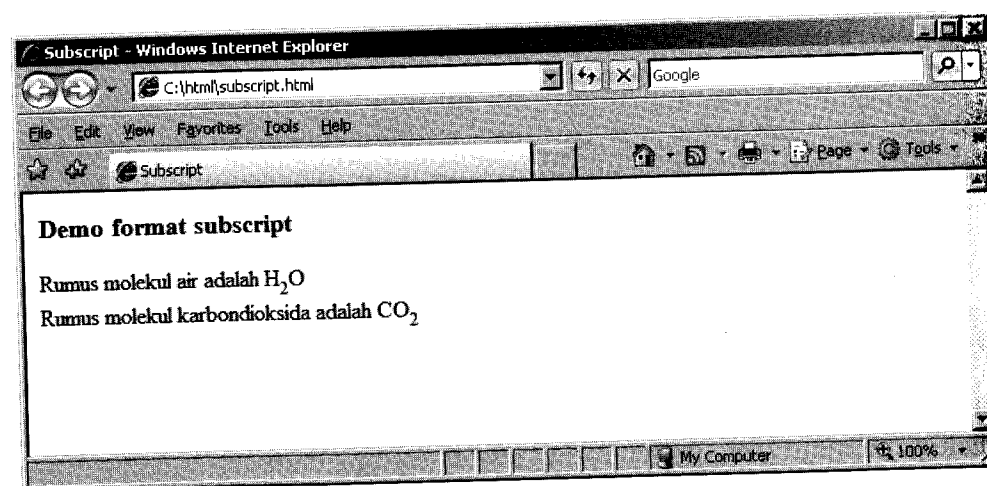
Format *subscript* merupakan kebalikan dari format *superscript*. Dalam format ini, teks akan ditempatkan pada posisi di bawah. Teks dengan format seperti ini pada umumnya kita jumpai pada saat membuat rumus kimia dari suatu molekul atau senyawa tertentu, misalnya H_2O , CO_2 , C_3H_8 , dan sebagainya. Dalam HTML, format *subscript* dibuat menggunakan tag `<sub>` dan `</sub>`. Berikut ini contoh kode yang akan menunjukkan cara kerja dari tag tersebut.

```
<html>
<head>
  <title>Subscript</title>
</head>
<body>

<h3>Demo format subscript</h3>

Rumus molekul air adalah H<sub>2</sub>O<br />
Rumus molekul karbondioksida adalah CO<sub>2</sub>

</body>
</html>
```



Membuat *Link*

Hampir semua halaman *web* yang kita jumpai di internet selalu menyediakan *link* ke dokumen atau halaman lain. *Link* (sebutan singkat untuk *hyperlink text*) berguna untuk memudahkan *user* dalam menelusuri seluruh isi atau informasi yang tersimpan dalam *website* bersangkutan. Secara *default*, *link* dalam dokumen HTML ditandai dengan teks berwarna biru dan memiliki garis bawah serta jika kita mengarahkan kursor *mouse* ke teks tersebut, maka penunjuk *mouse* akan berubah menjadi gambar tangan. Pada bagian ini kita akan mempelajari teknik-teknik pembuatan *link* yang dapat diterapkan dalam dokumen HTML.

Tag `<a>` (*anchor*)

Untuk pembuatan *link*, HTML menyediakan satu tag khusus, yaitu tag `<a>`, yang berpasangan dengan `</a>`. Setiap teks yang diapit oleh tag ini akan ditampilkan sebagai *link*. Bentuk umum pembuatan *link* dalam dokumen HTML adalah sebagai berikut:

```
<a href="DokumenLain">Teks Link</a>
```

Atribut `HREF` dalam tag `<a>` berfungsi untuk menentukan dokumen atau halaman mana yang akan dipanggil pada saat *link* tersebut diklik atau dipilih oleh *user*.

Jenis-jenis *Link*

HTML membedakan jenis *link* berdasarkan lokasi dokumen yang akan diakses. Adapun jenis-jenis tersebut adalah sebagai berikut:

- *Link* absolut
- *Link* relatif
- *Link* dalam dokumen yang sama

Link Absolut

Link absolut adalah suatu *link* yang tujuannya mengarah ke alamat *website* lain. Sebagai contoh, kita ingin mengembangkan *web* yang berisi tentang pemrograman *database*, lalu kita ingin menyediakan suatu *link* yang berkaitan dengan isi *website* kita (misalnya ke *website* Oracle), dengan menuliskan kode seperti berikut:

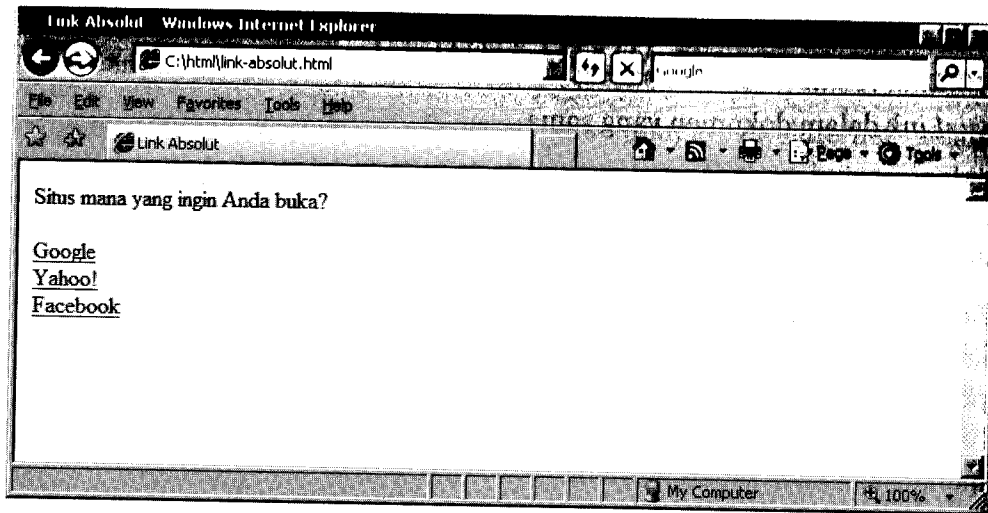
```
<a href="http://www.oracle.com">Oracle Website</a>
```

Link seperti di atas disebut sebagai *link* absolut. Di sini, alamat *website* tujuan harus ditulis secara lengkap.

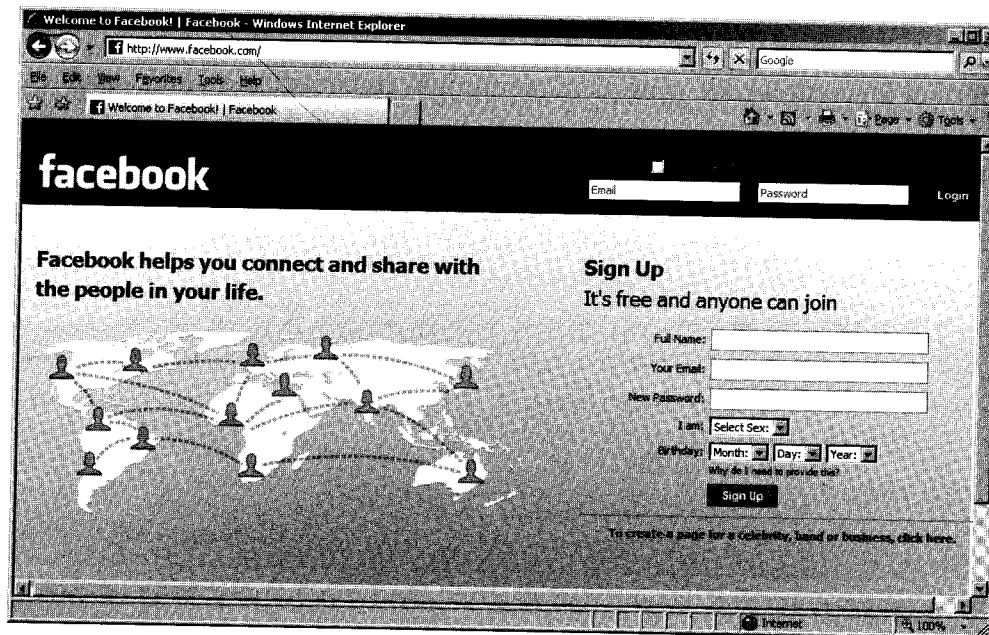
Berikut ini contoh kode yang akan menunjukkan penggunaan *link* absolut.

link-absolut.html

```
<html>
<head>
  <title>Link Absolut</title>
</head>
<body>
  <h3></h3>
  <p>Situs mana yang ingin Anda buka?</p>
  <a href="http://www.google.com">Google</a><br />
  <a href="http://www.google.com">Yahoo!</a><br />
  <a href="http://www.facebook.com">Facebook</a><br />
</body>
</html>
```



Sebagai contoh, jika user memilih **link Facebook**, maka *web browser* akan langsung mengarah ke situs <http://www.facebook.com>



Link Relatif

Link relatif adalah suatu *link* yang tujuannya mengarah ke dokumen-dokumen lain yang masih berada di dalam satu komputer yang sama. Penulisan alamat tujuannya pun tidak perlu ditulis secara lengkap, cukup nama dokumennya saja. Perhatikan contoh kode berikut:

Nama file: **index.html**

```
...
<a href="Kontak.html">Kontak</a>
...
```

Kode di atas digunakan untuk membuat *link* ke *file* Kontak.html yang lokasinya berada dalam satu *folder* dengan *file* index.html. Berikut ini contoh kode yang akan menunjukkan konsep di atas.

Index.html

```
<html>
<head>
  <title>Link HTML</title>
</head>
<body>
  <h3>Demo membuat link relatif</h3>
  Untuk informasi lebih lanjut, silahkan kontak kami
  <a href="kontak.html">di sini</a>.
</body>
</html>
```

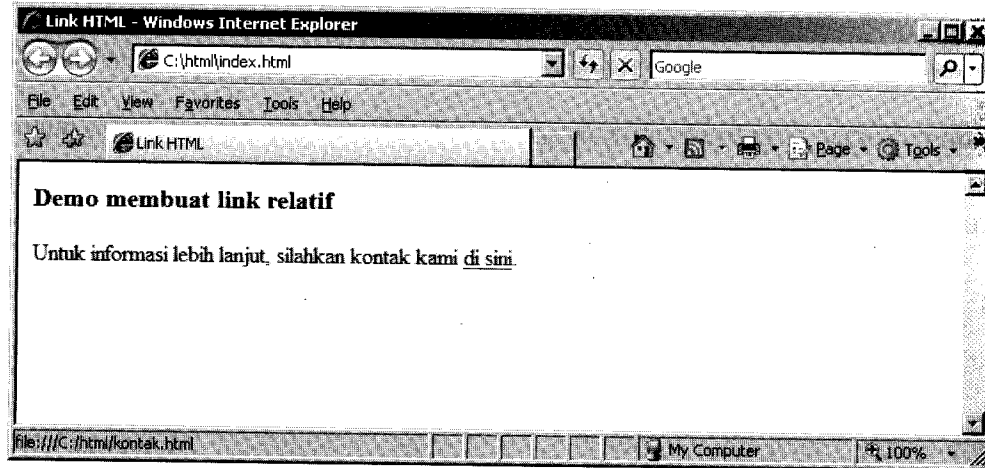
Kontak.html

```
<html>
<head>
  <title>Kontak</title>
</head>
<body>
  <p><strong>Kontak kami:</strong></p>
  <strong>PT. Multidimension Group</strong><br />
  Alamat: Jl. Kembar Baru Tenggara 234 No. 18-21 Bandung<br />
```

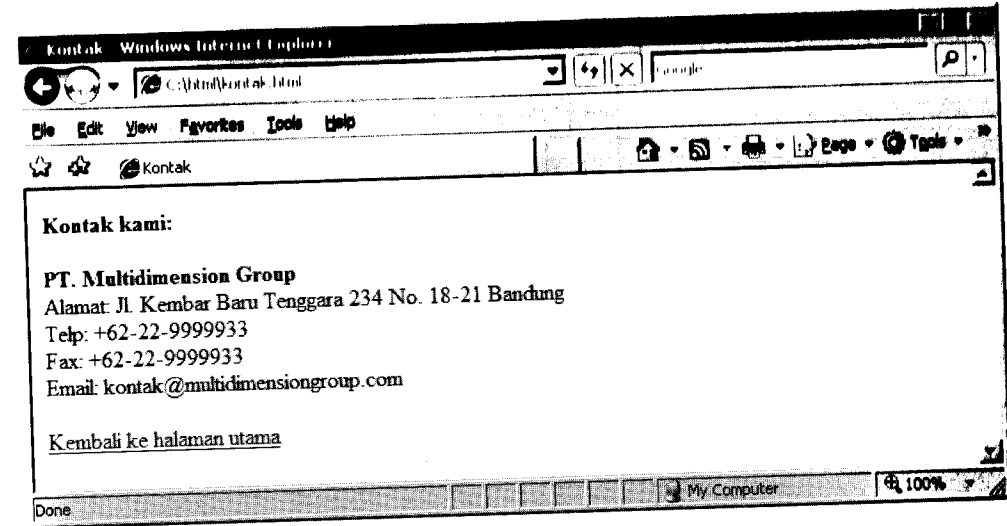
Telp: +62-22-9999933
Fax: +62-22-9999933
Email: kontak@multidimensiongroup.com

```
<p><a href="index.html">Kembali ke halaman utama</a></p>
</body>
</html>
```

Ketika *file* index.html dipanggil melalui suatu *web browser*, hasil yang akan ditampilkan adalah sebagai berikut:



Jika *link* dalam dokumen di atas dipilih atau diklik, maka tampilan akan berubah menjadi seperti berikut:



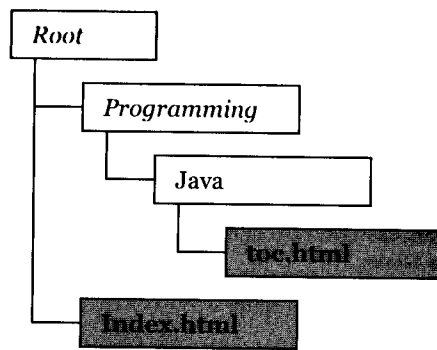
Link yang terdapat dalam dokumen kontak.html berguna untuk mengarah kembali ke dokumen awal.

Jika kita ingin mengakses dokumen lain yang berada dalam *subfolder* tertentu, gunakan kode berikut:

Nama *file*: **index.html**

```
...
<a href="programming/java/toc.html">Java Programming</a>
...
```

Programming adalah *subfolder* dari *folder* yang menyimpan *file* index.html dan **Java** adalah *subfolder* dari *folder* Programming. Jika digambarkan, posisi penempatan *folder* dan *file*-nya adalah sebagai berikut:



Keterangan:

Pada gambar di atas, teks yang dicetak tebal adalah nama file.

Link dalam Dokumen yang Sama

Link juga dapat berperan untuk mengaktifkan bagian teks atau gambar tertentu yang masih berada dalam satu dokumen yang sama. Link seperti ini biasanya dijumpai pada halaman web yang memiliki informasi yang panjang atau pada dokumen yang memiliki daftar isi di bagian atasnya. Untuk membuat link seperti ini, terlebih dahulu kita perlu memberikan nama pada bagian teks atau gambar tertentu sebagai bagian yang akan dituju ketika link tersebut diklik. Pemberian nama tersebut dilakukan dengan cara menuliskan kode berikut:

```
<a name="TujuanLink">Teks atau gambar</a>
```

Selanjutnya, untuk membuat link yang akan mengarah ke teks atau gambar yang telah dinamai tersebut adalah dengan cara seperti berikut:

```
<a href="#TujuanLink">Teks atau gambar</a>
```

Tanda # di atas menunjukkan bahwa tujuan link adalah nama bagian, bukan berupa file. Untuk membuktikan konsep di atas, silakan Anda perhatikan contoh kode berikut ini.

```

<html>
<head>
  <title>Link dalam Satu Dokumen</title>
</head>
<body>
  <h1 align="center"><a name="atas">Menangkarkan Burung Murai Batu</a></h1>

  <p>Burung murai batu adalah burung yang paling banyak digemari di kalangan
  penggemar burung berkicau. Burung ini mampu mengalunkan lagu atau melodi
  yang sangat merdu dan mampu menirukan suara-suara dari burung lain. Bentuk
  fisiknya yang proporsional dan ekor yang panjang juga menambah keindahan
  burung murai batu.</p>

  <p>Semakin tinggi permintaan pasar dari berbagai daerah di Indonesia
  menjadikan burung ini selalu diburu oleh para penangkap liar serta
  menjadikan harga burung ini melambung tinggi. Oleh karena itu, diperlukan
  adanya suatu usaha yang dapat membantu kelestariannya agar tidak mengalami
  kepunahan. Salah satu cara yang mulai banyak digalakkan di kalangan
  penggemar burung adalah dengan menangkarkan burung tersebut. Berikut ini
  cara melakukan penangkarnya.</p>

  <a href="#induk">Pemilihan Indukan</a><br />
  <a href="#penjodohan">Proses Penjodohan</a><br />
  <a href="#kandang">Kandang Penangkaran</a><br />
  <a href="#anakan">Memelihara Anakan</a>
  <br /><br />

  <h3><a name="induk">Pemilihan Indukan</a></h3>
  <p>Burung indukan yang akan ditangkarkan harus dipilih secara jeli, baik
  jantan maupun betinanya. Dari sisi fisik, burung harus sehat, lincah, tidak
  cacat, dan bentuk fisiknya proporsional. Dari segi suara, pilihlah burung
  yang bersuara merdu, keras, dan rajin berkicau.</p>

  <p align="right"><a href="#atas">Kembali ke atas...</a></p>

  <h3><a name="penjodohan">Proses Penjodohan</a></h3>
  <p>Penjodohan dilakukan dengan mengenalkan terlebih dahulu calon pasangan
  tersebut, yaitu melalui cara menempatkannya secara berdekatan dan terus
  menerus. Setelah keduanya terlihat saling tertarik yang ditandai dengan
  saling bersahut-sahutan, maka keduanya dapat disatukan dalam kandang
  penangkaran.</p>

  <p align="right"><a href="#atas">Kembali ke atas...</a></p>

```



```
<h3><a name="kandang">Kandang Penangkaran</a></h3>
```

```
<p>Ukuran kandang penangkaran tidak dapat dibakukan. Pada umumnya, para penangkar menggunakan ukuran 1m x 1m x 2m untuk p x l x t.</p> Bagian atas kandang sebaiknya menggunakan bahan yang tidak mudah panas sehingga suhu di dalam kandang dapat terkontrol. Kandang sebaiknya menghadap ke arah Timur karena untuk menyerap sinar matahari pagi, yang berguna untuk membuat burung menjadi sehat dan membunuh bibit penyakit di dalam kandang. Kandang harus dilengkapi dengan tempat makan dan minum, sarang, dan tenggeran. Agar lebih terkesan alami, dalam kandang juga dapat ditempatkan tumbuhan perdu yang memiliki banyak ranting, seperti beringin, kersen, dan lain-lain. Untuk menjaga kesehatan burung yang ditangkarkan, kebersihan kandang harus selalu terjaga.</p>
```

```
<p align="right"><a href="#atas">Kembali ke atas...</a></p>
```

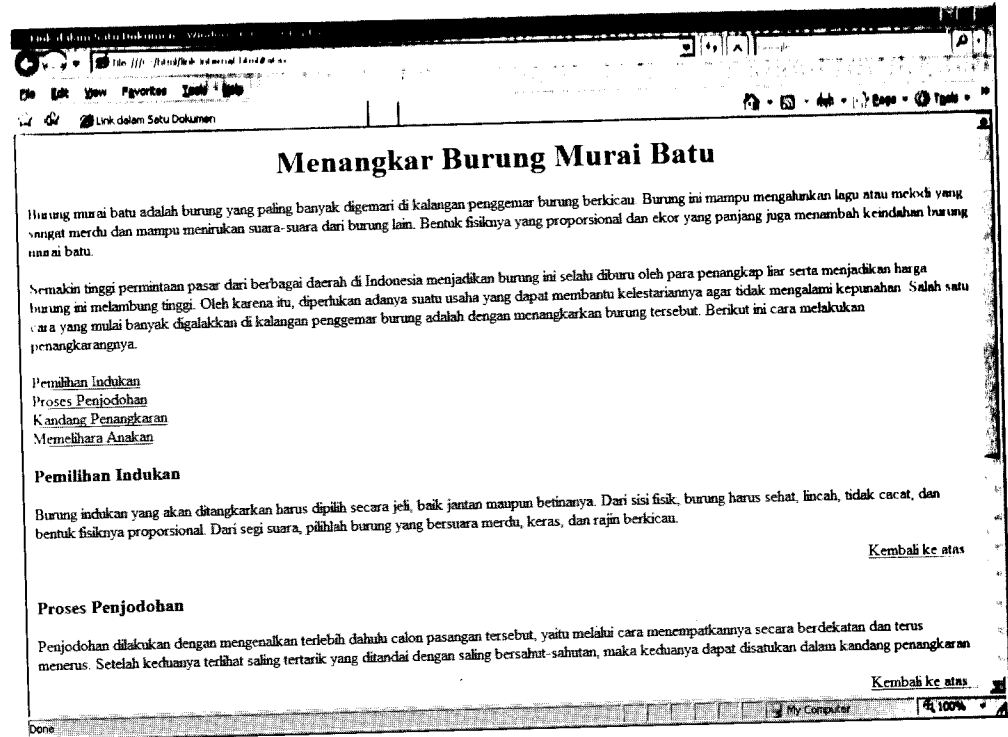
```
<h3><a name="anakan">Memelihara Anakan</a></h3>
```

```
<p>Setelah pasangan burung bertelur, mengeram, dan menetasakan anakan, kita harus mengambil alih proses perawatan anakan tersebut. Tujuannya adalah agar pasangan indukan tersebut dapat bereproduksi lagi dalam waktu singkat. Perawatan anakan harus dilakukan secara hati-hati karena pada usia 1-15 hari anakan burung murai batu mudah mati jika tidak memperoleh asupan gizi yang cukup.</p>
```

```
<p align="right"><a href="#atas">Kembali ke atas...</a></p>
```

```
</body>
```

```
</html>
```



Link ke Bagian Tertentu dalam Dokumen Lain

Link dapat juga mengarah ke suatu bagian tertentu yang terdapat dalam dokumen lain. Untuk membuat *link* semacam ini, bagian tertentu harus ditandai dengan pemberian nama seperti di atas. Yang berbeda hanyalah pada penulisan tujuan *link*-nya saja. Pada kasus seperti ini, kita perlu menambahkan nama bagian pada tujuan *link*, seperti berikut:

```
<a href="NamaFile#NamaBagian">Teks atau gambar</a>
```

Contoh:

```
<a href="Bab3.html#3.3">
  3.3. Link ke Bagian dalam File Lain
</a>
```

Kode di atas digunakan untuk membuat *link* yang mengarah ke dokumen **Bab3.html bagian 3.3**. Perhatikan contoh kode berikut ini.

File1.html

```
<html>
<head>
  <title>Link ke Bagian Tertentu dalam
    Dokumen Lain</title>
</head>
<body>

<h3>
  Demo membuat link ke bagian tertentu dalam dokumen lain.
</h3>

<a href="File2.html#bagian3">
  Pindah ke Bagian3 dalam File2.html
</a>

</body>
</html>
```

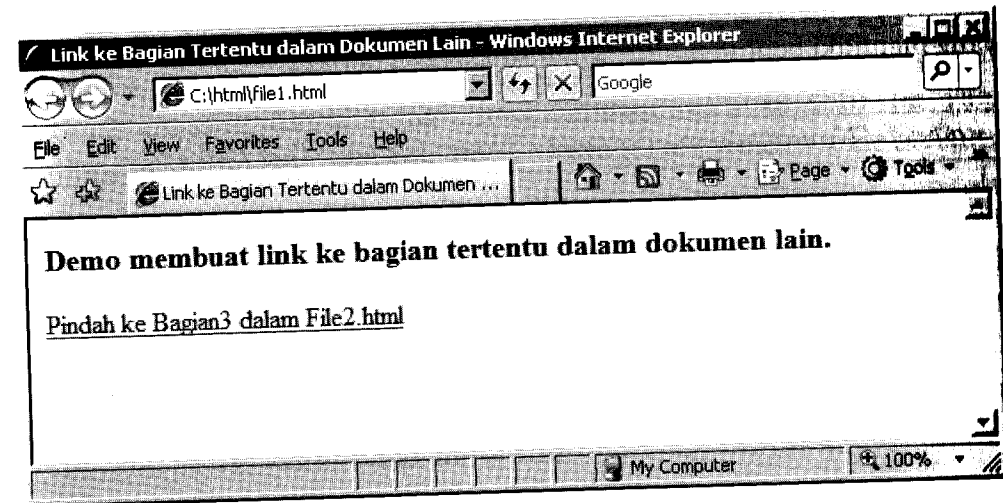
File2.html

```
<html>
<head>
  <title>Dokumen Lain</title>
</head>
<body>
<h3><a name="bagian1">Bagian 1</a></h3>
<p>Ini adalah teks yang terdapat pada Bagian 1. Ini adalah teks yang
terdapat pada Bagian 1. Ini adalah teks yang terdapat pada Bagian 1. Ini
adalah teks yang terdapat pada Bagian 1. Ini adalah teks yang terdapat pada
Bagian 1. Ini adalah teks yang terdapat pada Bagian 1. Ini adalah teks yang
terdapat pada Bagian 1...</p>

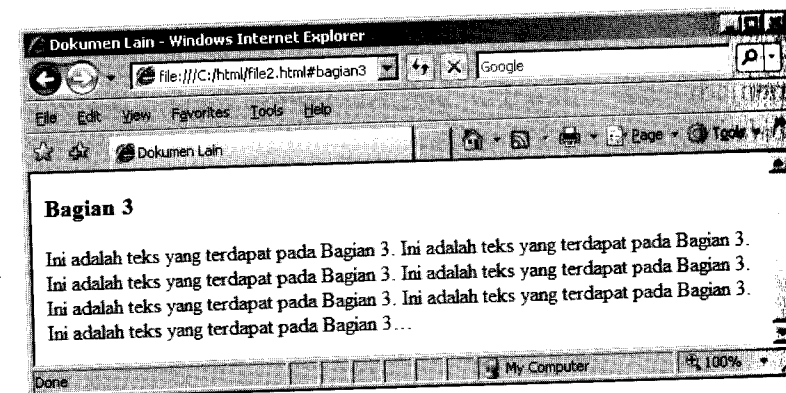
<h3><a name="bagian2">Bagian 2</a></h3>
<p>Ini adalah teks yang terdapat pada Bagian 2. Ini adalah teks yang
terdapat pada Bagian 2. Ini adalah teks yang terdapat pada Bagian 2. Ini
adalah teks yang terdapat pada Bagian 2. Ini adalah teks yang terdapat pada
Bagian 2. Ini adalah teks yang terdapat pada Bagian 2. Ini adalah teks yang
terdapat pada Bagian 2...</p>
```

```
<h3><a name="bagian3">Bagian 3</a></h3>
<p>Ini adalah teks yang terdapat pada Bagian 3. Ini adalah teks yang
terdapat pada Bagian 3. Ini adalah teks yang terdapat pada Bagian 3. Ini
adalah teks yang terdapat pada Bagian 3. Ini adalah teks yang terdapat pada
Bagian 3. Ini adalah teks yang terdapat pada Bagian 3. Ini adalah teks yang
terdapat pada Bagian 3...</p>
```

```
</body>
</html>
```



Jika *link* di atas diklik, maka *web browser* akan menampilkan hasil seperti gambar di samping ini.



Penentuan Target Link

Target atau tujuan dokumen yang dihubungkan melalui suatu *link* dapat ditentukan sendiri oleh kita. Target yang dimaksud di sini adalah tempat dimana dokumen akan ditempatkan atau ditampilkan. Dalam keadaan *default* (ketika atribut TARGET tidak disertakan), dokumen baru yang ditampilkan pada saat link dipilih akan menutup atau menimpa dokumen lama (dokumen yang mengandung *link* bersangkutan). Untuk melakukan hal ini, kita hanya perlu menambahkan atribut TARGET di dalam tag `<a>`, yang memiliki bentuk umum penulisan seperti berikut:

```
<a href="NamaDokumen" target="NamaTarget">Text Link</a>
```

NamaDokumen adalah nama dokumen yang akan ditampilkan jika *link* bersangkutan dipilih/diklik dan *NamaTarget* adalah target atau tempat tujuan dari dokumen yang akan ditampilkan. *NamaTarget* bisa bernilai TOP, _BLANK, maupun nama frame (jika layar *web* kita bagi dengan menggunakan frame).

Menampilkan Gambar

Gambar dapat dijadikan sebagai pemanis halaman *web*, membuat *web* menjadi lebih hidup dan menarik. Saat ini, jarang sekali kita jumpai suatu *website* yang tidak melibatkan gambar di dalamnya. Oleh karena itu, sebagai pemrogram *web*, Anda harus bisa menampilkan dan mengatur gambar dalam dokumen HTML yang Anda buat. Bagian ini akan menjelaskan bagaimana cara melakukan hal tersebut.

Tag (image)

Untuk menampilkan gambar di dalam dokumen HTML, kita perlu menggunakan tag `<img>`. Tag ini tidak memiliki pasangan *tag* sehingga perlu diakhiri atau ditutup dengan tanda slash (/). Bentuk umum penggunaan tag `<img>` adalah sebagai berikut:

```

```

NamaFileGambar adalah URL dari gambar yang ingin ditampilkan. Sebagai contoh, jika kita ingin menampilkan gambar **foto.jpg** yang terdapat pada direktori **images**, maka kita perlu menuliskan kode berikut:

```

```

Atribut SRC pada tag `<img>` di atas menunjukkan lokasi atau sumber gambar yang akan dimuat ke dalam dokumen.

Berikut ini contoh kode yang akan menunjukkan cara menampilkan gambar di dalam suatu dokumen HTML.

```
<html>
<head>
  <title>Menampilkan Gambar</title>
</head>
<body>

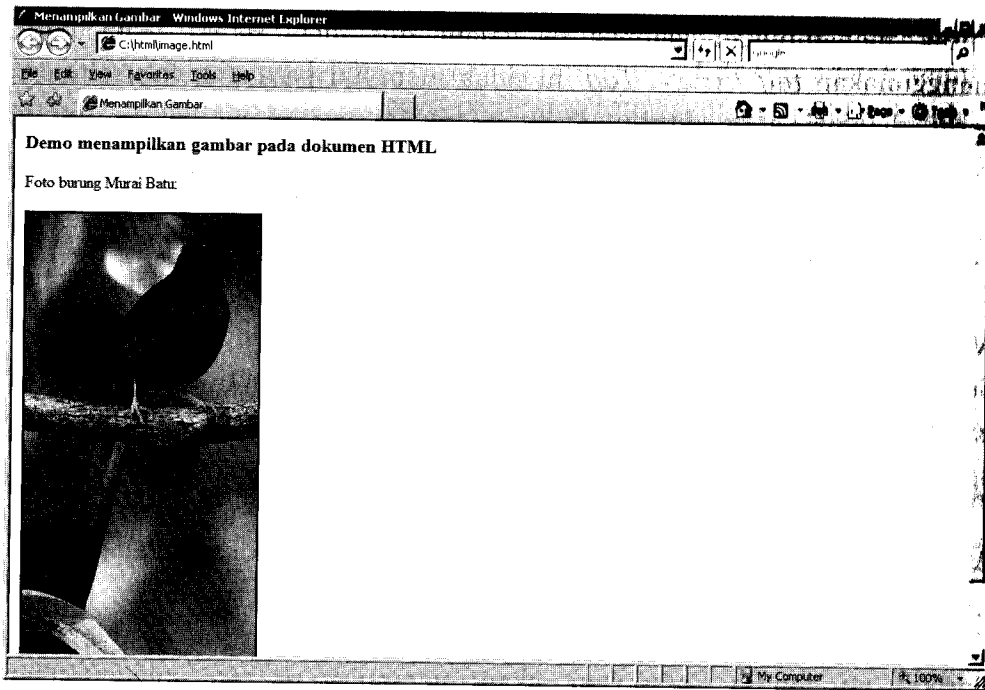
  <h3>Demo menampilkan gambar pada dokumen HTML</h3>

  Foto burung Murai Batu:<br /><br />

  <!-- menampilkan gambar -->
  

</body>
</html>
```

Hasil yang akan diberikan di halaman *web browser* adalah sebagai berikut



Mengatur Besar Kecilnya Gambar

Gambar yang ditampilkan pada dokumen HTML terkadang tidak memiliki ukuran yang sama dengan ukuran sebenarnya. Untuk menghasilkan gambar dengan kualitas yang mirip aslinya, penentuan ukuran juga menjadi hal yang penting untuk diperhatikan. Maka dari itu, perbandingannya harus sesuai dengan ukuran asli dari gambar tersebut. Untuk menentukan ukuran gambar, kita perlu mengisi atribut HEIGHT dan WIDTH yang terdapat pada *tag* . Atribut HEIGHT berfungsi untuk menentukan tinggi gambar dan WIDTH untuk menentukan lebar gambar. Sebagai contoh, perhatikan contoh kode berikut ini.

```

```

Kode di atas digunakan untuk menampilkan gambar dari *file* foto.jpg yang berada dalam direktori *images* dengan ukuran 200×180 piksel.

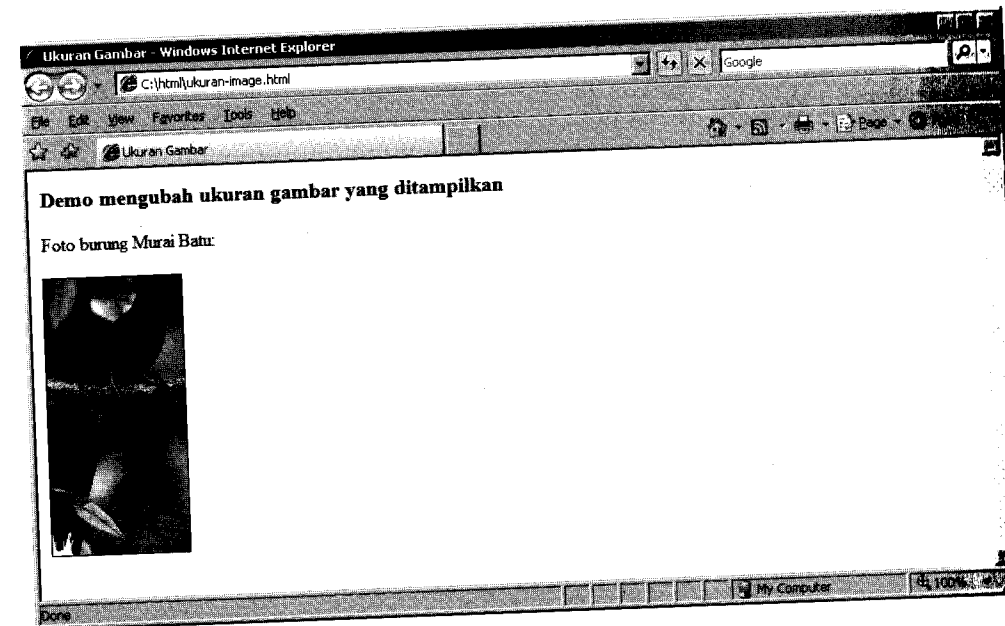
Berikut ini contoh kode yang akan menunjukkan cara mengecilkan ukuran gambar dalam dokumen HTML.

```
<html>
<head>
  <title>Ukuran Gambar</title>
</head>
<body>
  <h3>Demo mengubah ukuran gambar yang ditampilkan</h3>

  Foto burung Murai Batu:<br /><br />

  <!-- menampilkan gambar -->
  

</body>
</html>
```



Menambahkan Judul pada Gambar

Kita dapat menambahkan judul pada gambar yang ditampilkan dengan cara mengisi atribut `TITLE` yang terdapat dalam tag `<img>`. Bentuk umum penulisan atribut ini adalah sebagai berikut:

```

```

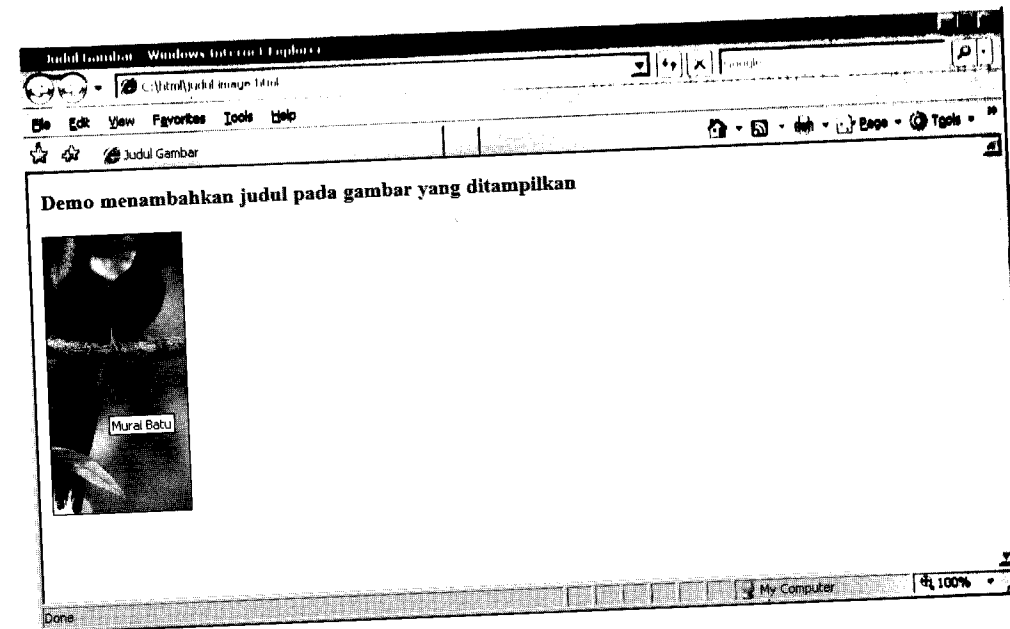
Contoh kode di bawah ini akan mendemonstrasikan cara penambahan judul pada gambar yang ingin ditampilkan.

```
<html>
<head>
  <title>Judul Gambar</title>
</head>
<body>

<h3>Demo menambahkan judul pada gambar
yang ditampilkan</h3>

<!-- menampilkan gambar -->


</body>
</html>
```



Judul akan ditampilkan sebagai teks *tooltip* pada saat kita mendekatkan penunjuk *mouse* ke arah gambar bersangkutan. Jika kita menggeser penunjuk *mouse* keluar gambar maka judul tersebut tidak akan muncul lagi. Teks *tooltip* semacam ini juga dapat dibuat menggunakan atribut `ALT`, seperti yang tampak pada contoh kode di bawah ini.

```
<html>
<head>
  <title>Judul Gambar</title>
</head>
<body>

<h3>Demo menambahkan judul pada gambar
yang ditampilkan</h3>

<!-- menampilkan gambar -->


</body>
</html>
```

Pada saat dijalankan, hasil yang diberikan oleh kode di atas akan sama seperti pada contoh kode sebelumnya.

Menggabungkan Gambar dan Teks

Jika kita ingin menggabungkan atau menyisipkan gambar di dalam suatu teks atau paragraf tertentu, terdapat pilihan posisi gambar yang harus diperhatikan:

Secara vertikal:

- Teks berada sejajar dengan bagian atas gambar
- Teks berada sejajar dengan bagian tengah gambar
- Teks berada sejajar dengan bagian bawah gambar

Secara horisontal:

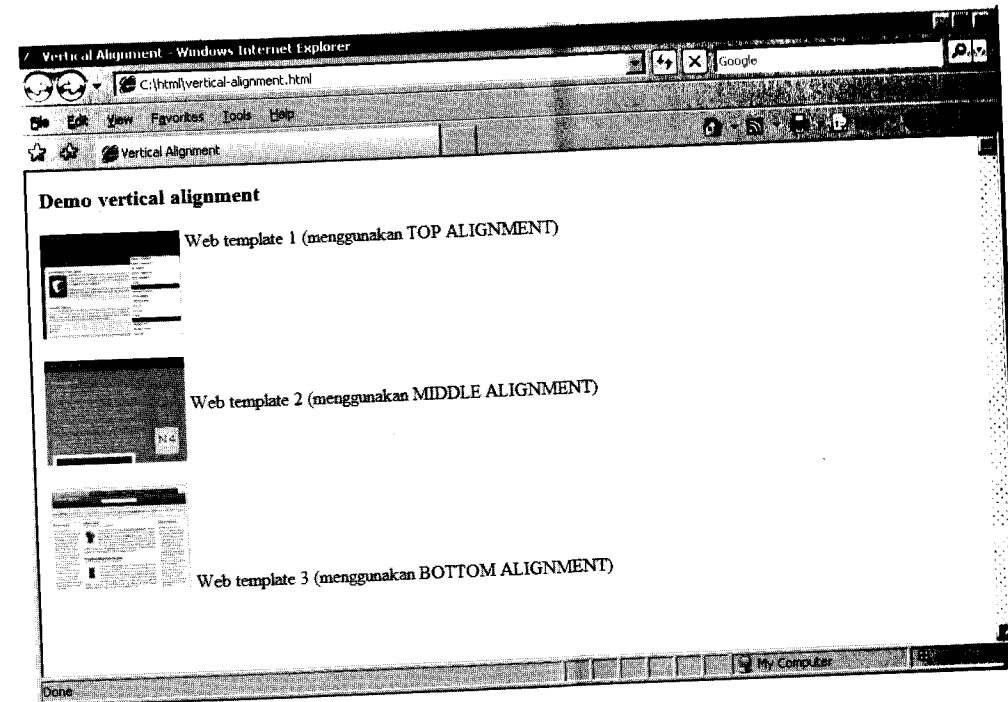
- Gambar berada di bagian kiri teks/paragraf
- Gambar berada di bagian kanan teks/paragraf

Untuk posisi gambar yang dilihat dari arah vertikal dapat ditentukan dengan mengisi nilai TOP, MIDDLE, atau BOTTOM ke dalam atribut ALIGN yang terdapat pada tag . Berikut ini contoh kode yang akan menunjukkan perbedaan dari ketiga nilai tersebut.

```
<html>
<head>
  <title>Vertical Alignment</title>
</head>
<body>
<h3>Demo vertical alignment</h3>

<p>
Web template 1 (menggunakan TOP ALIGNMENT)</p>
<p>
Web template 2 (menggunakan MIDDLE ALIGNMENT)</p>
<p>
Web template 3 (menggunakan BOTTOM ALIGNMENT)</p>
```

```
</body>
</html>
```



Adapun untuk posisi gambar yang dilihat dari arah horizontal ditentukan dengan mengisi nilai LEFT dan RIGHT ke dalam atribut ALIGN pada tag . Dengan mengisi atribut tersebut dengan nilai LEFT maka gambar akan berada di sebelah kiri teks. Sebaliknya, nilai RIGHT akan menyebabkan posisi gambar akan berada di sebelah kanan teks. Berikut ini contoh kode yang menunjukkan konsep tersebut.

```
<html>
<head>
  <title>Horizontal Alignment</title>
</head>
<body>
<h3>Demo horizontal alignment</h3>
<p>
```

```

```

Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web. Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web. Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web. Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web. Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web. Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web. Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web. Web template 1 ini terdiri atas dua kolom dengan menu berada di sisi kanan halaman web.

```
</p>
```

```
<p>
```

```

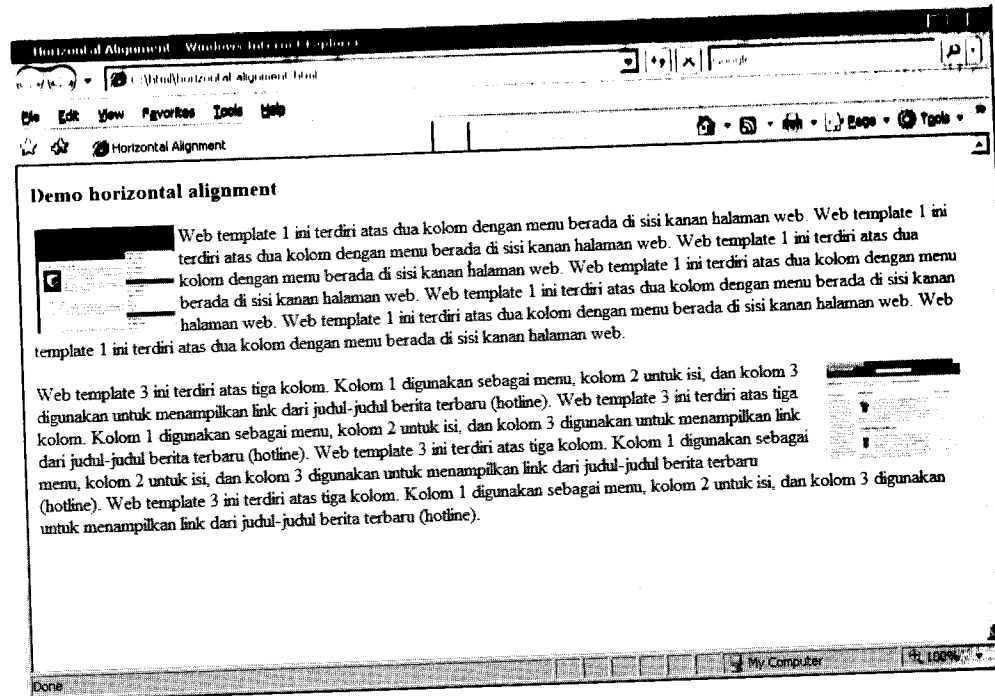
```

Web template 3 ini terdiri atas tiga kolom. Kolom 1 digunakan sebagai menu, kolom 2 untuk isi, dan kolom 3 digunakan untuk menampilkan link dari judul-judul berita terbaru (hotline). Web template 3 ini terdiri atas tiga kolom. Kolom 1 digunakan sebagai menu, kolom 2 untuk isi, dan kolom 3 digunakan untuk menampilkan link dari judul-judul berita terbaru (hotline). Web template 3 ini terdiri atas tiga kolom. Kolom 1 digunakan sebagai menu, kolom 2 untuk isi, dan kolom 3 digunakan untuk menampilkan link dari judul-judul berita terbaru (hotline). Web template 3 ini terdiri atas tiga kolom. Kolom 1 digunakan sebagai menu, kolom 2 untuk isi, dan kolom 3 digunakan untuk menampilkan link dari judul-judul berita terbaru (hotline). Web template 3 ini terdiri atas tiga kolom. Kolom 1 digunakan sebagai menu, kolom 2 untuk isi, dan kolom 3 digunakan untuk menampilkan link dari judul-judul berita terbaru (hotline).

```
</p>
```

```
</body>
```

```
</html>
```



Menjadikan Gambar sebagai Link

Seperti telah disinggung sebelumnya bahwa suatu *link* dalam dokumen HTML dapat dibuat menggunakan teks maupun gambar. Pada bagian ini, kita akan mempelajari bagaimana membuat *link* yang berupa gambar.

Untuk menjadikan gambar sebagai suatu *link* kita hanya tinggal menempatkan posisi *tag* `<img>` di dalam *tag* `<a>`. Bentuk umum penulisan kodenya adalah seperti berikut:

```
<a href="DokumenTujuan"></a>
```

Link berupa gambar banyak dijumpai dalam dokumen gambar yang diperkecil (*thumbnail*). Namun, ketika *link* gambar tersebut diklik, maka

web browser akan menampilkan dokumen berupa gambar dalam ukuran sebenarnya.

Contoh kode berikut ini akan menunjukkan konsep di atas.

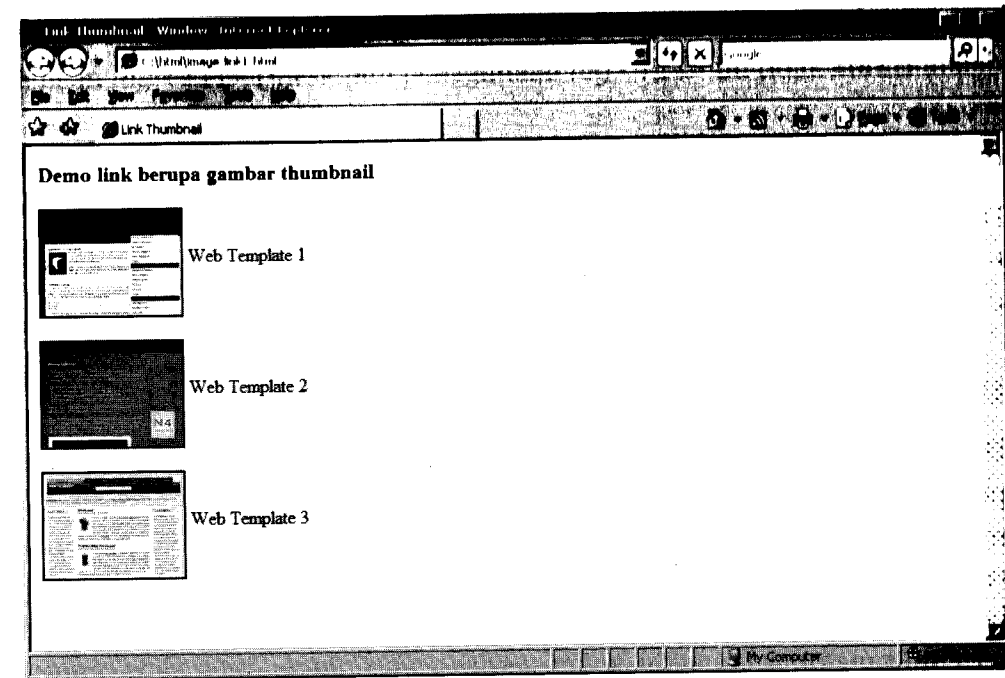
```
<html>
<head>
  <title>Link Thumbnail</title>
</head>
<body>
<h3>Demo link berupa gambar thumbnail</h3>

<p>
<a href="wt1aktual.gif"><a>
Web Template 1
</p>

<p>
<a href="wt2aktual.gif"><a>
Web Template 2
</p>

<p>
<a href="wt3aktual.gif"><a>
Web Template 3
</p>

</body>
</html>
```



Jika salah satu gambar di atas diklik maka *web browser* akan menjadi dokumen tujuan dari *link* tersebut.

Selain untuk tujuan seperti di atas, *link* berupa gambar juga banyak dijumpai pada gambar-gambar iklan atau sponsor dari *web-web* tertentu. Pada umumnya, pada iklan tersebut link akan disajikan dalam bentuk logo perusahaan, dan ketika logo tersebut diklik maka *web browser* akan menampilkan *website* resmi dari perusahaan bersangkutan.

Membuat Daftar (*List*)

Dalam pembuatan dokumen HTML, ada kalanya kita perlu membuat suatu daftar (*list*) atau kumpulan *item* tertentu. Daftar *item* tersebut dapat ditulis secara berurutan atau tidak. Bagian ini akan mempelajari cara membuat

jenis-jenis daftar tersebut menggunakan *tag-tag* yang telah disediakan dalam HTML.

Jenis Daftar

Dalam HTML, daftar dibedakan menjadi dua, yaitu:

- Daftar yang terurut
- Daftar yang tidak terurut

Daftar yang Terurut

Daftar yang terurut biasanya ditandai dengan penggunaan aturan penomoran tertentu. Penomoran ini bisa menggunakan angka (1, 2, 3, ...) maupun karakter alfabet tertentu (a, b, c, ... atau i, ii, iii, ...). Daftar yang terurut pada umumnya digunakan untuk *item-item* yang saling berhubungan satu sama lain, atau untuk menuliskan langkah-langkah tertentu yang tidak dapat dilakukan secara acak. Contoh daftar yang terurut adalah seperti pada tata-cara memasak atau membuat kue tertentu, misalnya:

Cara memasak mie instant:

1. Rebus air sebanyak 200cc sampai mendidih
2. Masukkan 1 butir telur ayam
3. Masukkan mie instant 1 bungkus
4. Masukkan sayuran yang telah dipotong-potong
5. Masukkan bumbu dan aduk hingga rata, lalu biarkan sekitar 3 menit
6. Angkat dan masakan siap dihidangkan

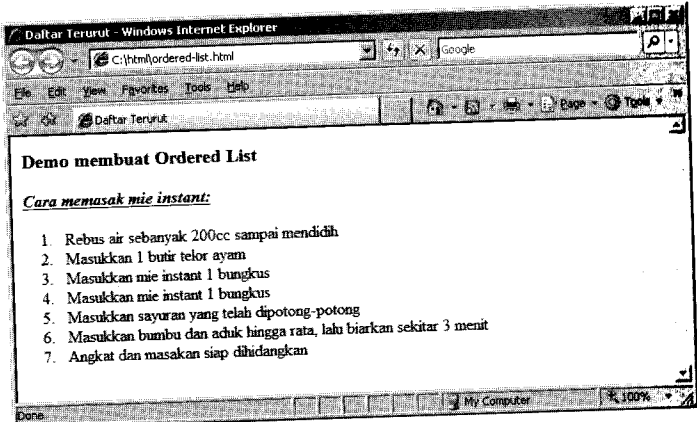
Untuk membuat suatu daftar yang terurut dalam dokumen HTML, kita perlu menggunakan *tag* `<ul>` (*unordered list*), yang berpasangan dengan *tag* `</ul>`. *Tag* `<ul>` digunakan untuk memulai suatu daftar terurut tertentu, sedangkan *tag* `</ul>` berfungsi untuk menandakan bahwa daftar tersebut telah berakhir. Masing-masing *item* di dalam daftar tersebut harus

dibuat menggunakan *tag* `<li>` (*list item*), yang kemudian perlu ditutup dengan *tag* `</li>` (meskipun penggunaan *tag* `</li>` ini tidak mutlak).

Berikut ini contoh sederhana yang akan menunjukkan cara membuat daftar terurut dalam dokumen HTML.

```
<html>
<head>
  <title>Daftar Terurut</title>
</head>
<body>
  <h3>Demo membuat Ordered List</h3>
  <strong><i><u>Cara memasak mie instant:</u></i></strong>
  <ol>
    <li>Rebus air sebanyak 200cc sampai mendidih</li>
    <li>Masukkan 1 butir telur ayam</li>
    <li>Masukkan mie instant 1 bungkus</li>
    <li>Masukkan mie instant 1 bungkus</li>
    <li>Masukkan sayuran yang telah dipotong-potong</li>
    <li>Masukkan bumbu dan aduk hingga rata,
      lalu biarkan sekitar 3 menit</li>
    <li>Angkat dan masakan siap dihidangkan</li>
  </ol>
</body>
</html>
```

Dalam keadaan *default*, penomoran pada tag `<ol>` akan menggunakan angka 1, 2, 3, dst. (seperti yang tampak pada contoh di atas). Meskipun demikian,



kita bisa saja menggantinya dengan karakter lain yang juga ditulis secara berurutan. Caranya adalah dengan menyertakan atribut TYPE pada tag `<ol>`. Berikut ini nilai yang diizinkan untuk mengisi atribut tersebut.

- A: akan membuat daftar dengan jenis penomoran berupa A, B, C, ...
- a: akan membuat daftar dengan jenis penomoran berupa a, b, c, ...
- I: akan membuat daftar dengan jenis penomoran berupa I, II, III, ...
- i: akan membuat daftar dengan jenis penomoran berupa i, ii, iii, ...

Contoh kode di bawah ini akan menunjukkan penggunaan masing-masing tipe penomoran di atas.

```
<html>
<head>
  <title>Tipe Penomoran pada OL</title>
</head>
<body>

<h3>Demo tipe penomoran pada tag <ol></h3>

<p>Menggunakan tipe A, B, C, ...</p>
<ol type="A">
  <li>Jalankan aplikasi Text Editor</li>
  <li>Tulis kode HTML dan simpan dengan ekstensi .htm
    atau .html</li>
  <li>Jalankan file HTML melalui web browser</li>
</ol>

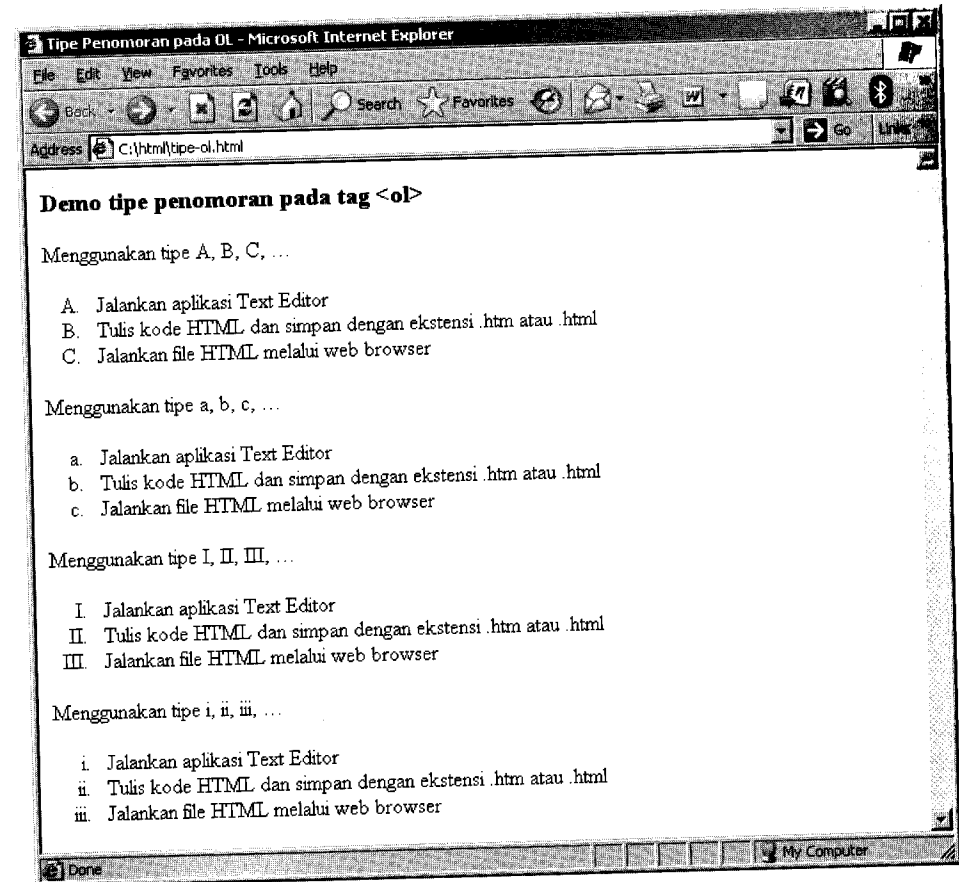
<p>Menggunakan tipe a, b, c, ...</p>
<ol type="a">
  <li>Jalankan aplikasi Text Editor</li>
  <li>Tulis kode HTML dan simpan dengan ekstensi .htm
    atau .html</li>
  <li>Jalankan file HTML melalui web browser</li>
</ol>

<p>Menggunakan tipe I, II, III, ...</p>
<ol type="I">
  <li>Jalankan aplikasi Text Editor</li>
  <li>Tulis kode HTML dan simpan dengan ekstensi .htm
```

```
    atau .html</li>
  <li>Jalankan file HTML melalui web browser</li>
</ol>

<p>Menggunakan tipe i, ii, iii, ...</p>
<ol type="i">
  <li>Jalankan aplikasi Text Editor</li>
  <li>Tulis kode HTML dan simpan dengan ekstensi .htm
    atau .html</li>
  <li>Jalankan file HTML melalui web browser</li>
</ol>

</body>
</html>
```



Daftar yang Tidak Terurut

Daftar yang tidak terurut adalah daftar yang *item-item*-nya dapat diubah posisinya secara acak. Daftar seperti ini tidak menggunakan penomoran seperti di atas, melainkan hanya dengan menggunakan tanda atau simbol tertentu, misalnya gambar kotak atau bulat. HTML menyediakan tag `<ul>` (*unordered list*), yang berpasangan dengan `</ul>`. Tag tersebut digunakan untuk membuat daftar tak terurut dalam dokumen HTML. Untuk pendefinisian masing-masing itemnya sama seperti pada tag `<ol>`, yaitu dengan menggunakan tag `<li>` dan `</li>`.

Berikut ini contoh kode yang akan menunjukkan cara pembuatan daftar tak terurut.

```
<html>
<head>
  <title>Daftar Tidak Terurut</title>
</head>
<body>

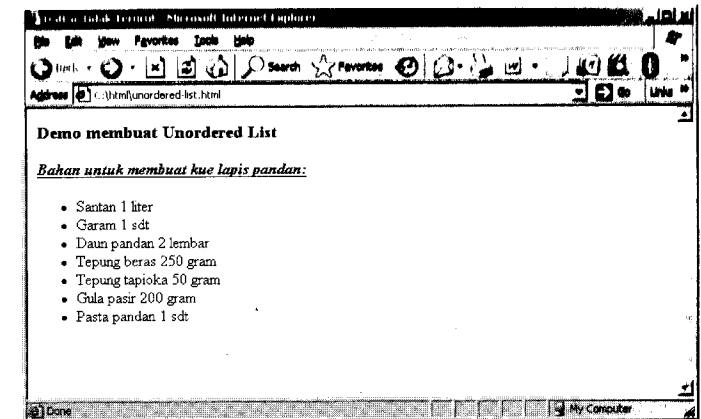
<h3>Demo membuat Unordered List</h3>

<p><strong><i><u>Bahan untuk membuat kue lapis
pandan:</u></i></strong></p>

<ul>
  <li>Santan 1 liter</li>
  <li>Garam 1 sdt</li>
  <li>Daun pandan 2 lembar</li>
  <li>Tepung beras 250 gram</li>
  <li>Tepung tapioka 50 gram</li>
  <li>Gula pasir 200 gram</li>
  <li>Pasta pandan 1 sdt</li>
</ul>

</body>
</html>
```

Pada daftar di samping, tidak ada ketentuan yang mewajibkan bahwa *item-item*-nya harus ditulis secara berurutan. Artinya, kita bisa saja menulisnya secara acak karena tidak berkaitan dengan suatu prosedur atau langkah-langkah tertentu.



Tampak dari hasil yang diberikan pada contoh di atas bahwa dalam keadaan *default* tanda yang digunakan untuk setiap item pada tag `<ul>` adalah tanda bulat. Meskipun demikian, kita bisa saja mengubahnya dengan menyertakan atribut `TYPE` pada tag tersebut, sama halnya pada saat kita mengubah tipe penomoran pada tag `<ol>`. Perbedaannya terletak pada nilai yang diisikan untuk atribut `TYPE`. Dalam tag `<ul>`, nilai-nilai yang diizinkan untuk dimasukkan ke dalam atribut `TYPE` adalah sebagai berikut:

- `DISK`: gambar bulat yang diarsir (*default*)
- `CIRCLE`: gambar bulat yang tidak diarsir
- `SQUARE`: gambar kotak

Contoh kode di bawah ini akan menunjukkan cara pembuatan daftar menggunakan tipe-tipe di atas.

```
<html>
<head>
  <title>Tipe Unordered List</title>
</head>
<body>
```

<h3>Demo tipe bullet pada tag </h3>

<p>Menggunakan tipe DISK:</p>

```
<ul type="disk">
  <li>Apel</li>
  <li>Jeruk</li>
  <li>Pisang</li>
  <li>Melon</li>
  <li>Sawo</li>
</ul>
```

<p>Menggunakan tipe CIRCLE:</p>

```
<ul type="circle">
  <li>Apel</li>
  <li>Jeruk</li>
  <li>Pisang</li>
  <li>Melon</li>
  <li>Sawo</li>
</ul>
```

<p>Menggunakan tipe SQUARE:</p>

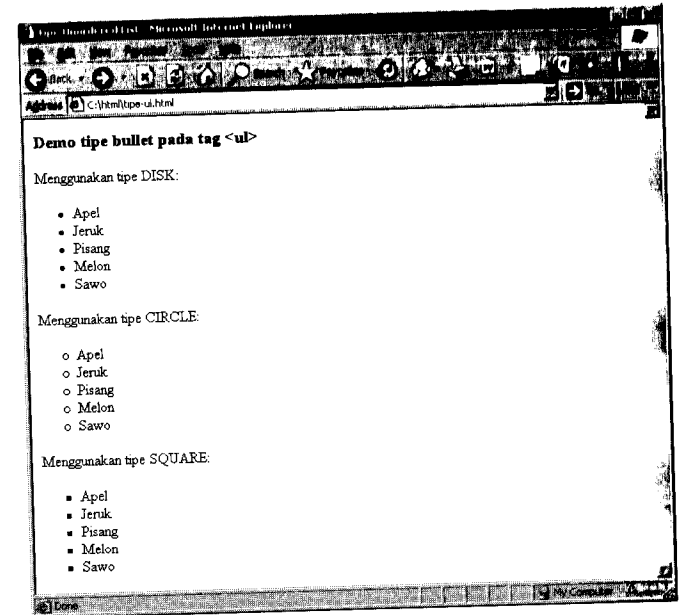
```
<ul type="square">
  <li>Apel</li>
  <li>Jeruk</li>
  <li>Pisang</li>
  <li>Melon</li>
  <li>Sawo</li>
</ul>
```

</body>

</html>

Catatan:

Gambar bullet di samping dapat diubah menggunakan gambar dari suatu file tertentu dengan menggunakan CSS (CSS tidak dibahas di dalam modul ini).



Daftar Bersarang (Nested List)

Dalam pembuatan dokumen HTML, ada kalanya kita dituntut untuk membuat suatu daftar yang bersarang (*item* maupun *subitem* memiliki daftar lagi). Daftar semacam ini dikenal dengan daftar bersarang (*nested list*). Untuk melakukan hal ini, kita perlu menyertakan tag atau di dalam *item* bersangkutan. Sebagai contoh, perhatikan kode di bawah ini.

```
<html>
<head>
  <title>Daftar Bersarang</title>
</head>
<body>

<h3>Demo membuat daftar bersarang (<i>nested list</i>)</h3>
<p><strong>Membuat Daftar</strong></p>
<ol>
  <li>Jenis Daftar dalam HTML
    <ul>
      <li>Daftar yang Terurut</li>
```



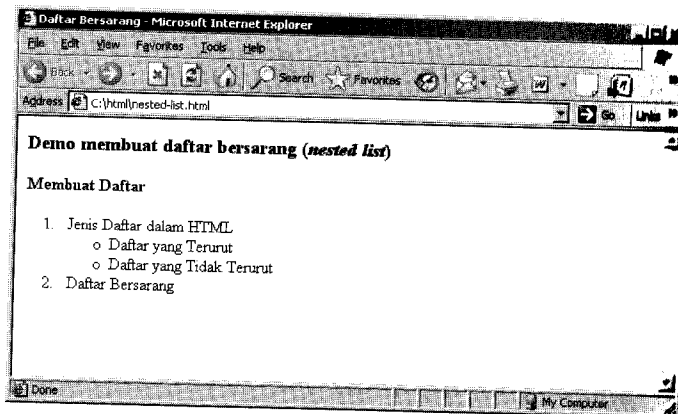
```

<li>Daftar yang Tidak Terurut</li>
</ul>
</li>
<li>Daftar Bersarang</li>
</ol>

</body>
</html>

```

Pada contoh ini, *item* 1 dan 2 adalah daftar. Akan tetapi, *item* 1 juga memiliki daftar lagi. *Item* dalam daftar bagian dalam juga dapat memiliki daftar yang lain lagi (bila diperlukan).



Definition List

Definition list adalah jenis daftar yang telah didefinisikan oleh HTML (*predefined*). Jenis daftar ini dibuat menggunakan *tag* `<dl>` dan `</dl>`. Untuk pembuatan *item-item*-nya, kita tidak menggunakan *tag* `<li>` dan `</li>`, melainkan `<dd>` dan `</dd>`. *Item-item* yang dibuat menggunakan *definition list* tidak mengandung penomoran yang terurut, melainkan akan ditulis menggunakan tanda bulat, sama seperti pada saat kita membuatnya menggunakan *tag* `<ul>` (dengan `TYPE=CIRCLE`). Selanjutnya, jika *item* tersebut memiliki sub atau anak lagi maka kita perlu membuatnya dengan menggunakan *tag* `<dt>` dan `</dt>`. Berikut ini contoh kode yang akan menunjukkan pembuatan daftar menggunakan *definition list*.

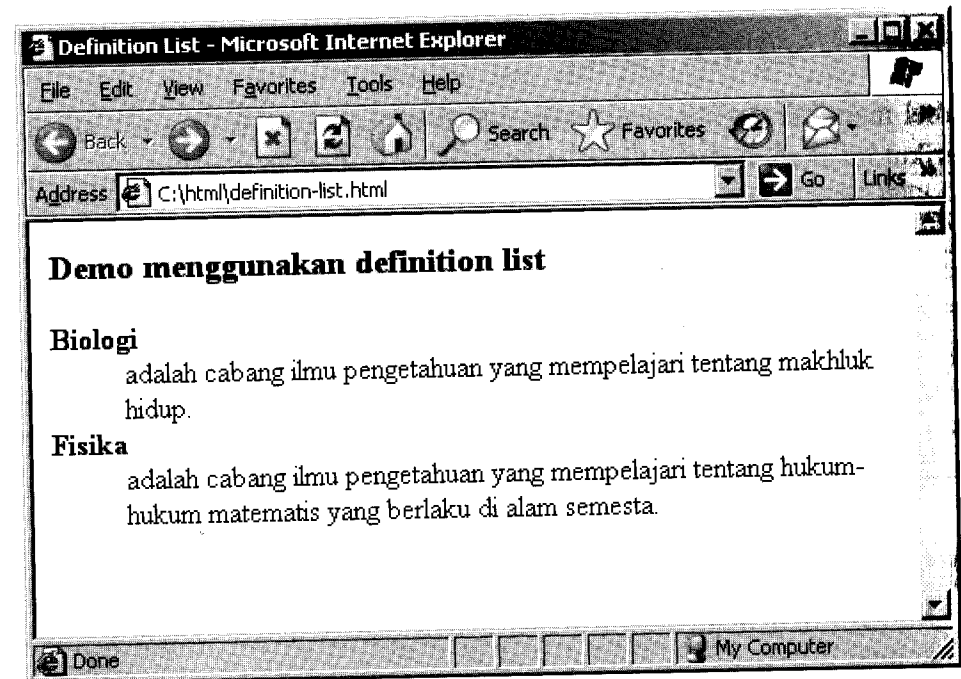
```

<html>
<head>
  <title>Definition List</title>
</head>
<body>

<h3>Demo menggunakan definition list</h3>
<dl>
  <dt><strong>Biologi</strong></dt>
  <dd>adalah cabang ilmu pengetahuan yang mempelajari
    tentang makhluk hidup.</dd>
  <dt><strong>Fisika</strong></dt>
  <dd>adalah cabang ilmu pengetahuan yang mempelajari
    tentang hukum-hukum matematis yang berlaku
    di alam semesta.</dd>
</dl>

</body>
</html>

```



Membuat Tabel

Dalam HTML, tabel memiliki dua fungsi:

- Untuk mengatur informasi ke dalam bentuk tabular
- Untuk mengatur tampilan (*layout*) dari suatu halaman atau bagian halaman tertentu

Sejak kemunculan CSS, fungsi kedua menjadi hal yang tidak direkomendasikan karena tabel hanya ditujukan untuk menampilkan data dalam bentuk tabular, sedangkan tampilan (*layout*) sebaiknya dibuat menggunakan CSS. Oleh karena itu, dalam modul ini tidak akan dibahas mengenai penggunaan tabel yang ditujukan untuk membuat tampilan layar.

Bagian-bagian Tabel

Tabel memiliki bagian-bagian yang disebut kolom, baris, dan sel. Gambar berikut ini akan menunjukkan letak dari bagian-bagian tersebut.

NIM	Nama	Alamat	Kota	Telepon
1208001	Hera Prakoso	Jl. Muri Batu 12	Semarang	024 8000341
1208002	Jamalludin	Jl. Muri Batu 33	Semarang	024 8034445
1208003	Arya Kusuma	Jl. Cusakawa 1	Jakarta	021 8999993
1208004	Darmawan	Jl. Haur 45	Bandung	022 7744455

Diagram labels: kolom (points to first column), baris (points to first row), sel (points to a cell in the last row).

Keterangan:

- Kolom merupakan atribut tabel. Setiap tabel dapat memiliki beberapa kolom.
- Baris merupakan isi sebuah *record* dari tabel bersangkutan.

- Sel adalah bagian terkecil dari suatu tabel yang berisi data atau nilai dari kolom dan baris tertentu.

Tag HTML untuk Pembuatan Tabel

Untuk keperluan pembuatan tabel di dalam dokumen, HTML menyediakan beberapa buah *tag* yang telah memiliki fungsi khusus. Semua *tag* yang disediakan tersebut memiliki *tag* pasangan, misalnya `<table>` akan berpasangan dengan `</table>`. Tabel berikut ini akan menunjukkan daftar *tag* yang dimaksud.

Tag	Keterangan
<code><table>...</table></code>	Tag ini digunakan untuk mendefinisikan tabel di dalam dokumen HTML
<code><caption>...</caption></code>	Tag ini digunakan untuk menyertakan judul tabel. Posisi judul tabel dapat diubah pada bagian atas atau bawah tabel, yaitu dengan mengubah atribut <code>ALIGN</code> menjadi <code>TOP</code> atau <code>BOTTOM</code>
<code><tr>...</tr></code>	Tag ini digunakan untuk membuat baris baru di dalam tabel
<code><th>...</th></code>	Tag ini digunakan untuk membuat sel <i>header</i> (kepala) atau judul kolom dari suatu tabel
<code><td>...</td></code>	Tag ini digunakan untuk membuat sel dalam tabel. Semua data yang dimunculkan dalam suatu tabel harus dibuat menggunakan tag <code><td></code> , yang berpasangan dengan tag <code></td></code> .

Berikut ini contoh kode yang akan menunjukkan penggunaan *tag-tag* di atas.

```
<html>
<head>
  <title>Tabel dalam HTML</title>
</head>
<body>
```

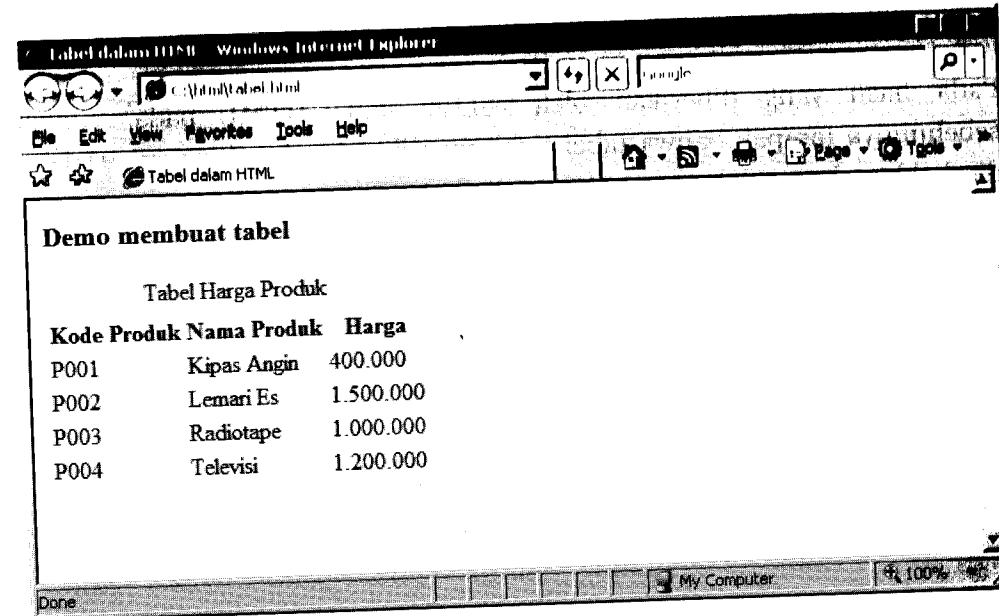


```
</head>
<body>
```

```
<h3>Demo membuat tabel</h3>
```

```
<table>
  <caption>Tabel Harga Produk</caption>
  <tr>
    <th>Kode Produk</th>
    <th>Nama Produk</th>
    <th>Harga</th>
  </tr>
  <tr>
    <td>P001</td>
    <td>Kipas Angin</td>
    <td>400.000</td>
  </tr>
  <tr>
    <td>P002</td>
    <td>Lemari Es</td>
    <td>1.500.000</td>
  </tr>
  <tr>
    <td>P003</td>
    <td>Radiotape</td>
    <td>1.000.000</td>
  </tr>
  <tr>
    <td>P004</td>
    <td>Televisi</td>
    <td>1.200.000</td>
  </tr>
</table>

</body>
</html>
```



Perhatikan kode berikut:

```
<tr>
  <th>Kode Produk</th>
  <th>Nama Produk</th>
  <th>Harga</th>
</tr>
```

Kode di atas digunakan untuk membuat satu baris dengan tiga buah kolom. Karena kolom-kolom dalam baris yang dibuat tersebut akan digunakan sebagai kolom judul, maka kita perlu menggunakan *tag* `<th>` (*table header*). Untuk baris-baris selanjutnya, kolomnya tidak dibuat menggunakan *tag* `<th>` lagi, melainkan *tag* `<td>`, karena merupakan sel normal (bukan judul), misalnya:

```
<tr>
  <td>P001</td>
  <td>Kipas Angin</td>
  <td>400.000</td>
</tr>
```

Menentukan Border Tabel

Pada hasil yang diberikan oleh contoh kode HTML pada subbab sebelumnya tampak bahwa tabel tidak memiliki *border* (bingkai). Hal ini disebabkan kita tidak menyertakan atribut BORDER pada tag `<table>`, yang berarti nilai dari atribut tersebut akan diisi 0 (nol). Agar *border* terlihat pada masing-masing sel tabel, kita perlu mengisi nilai lebih besar dari 0. Semakin tinggi nilai yang kita masukkan, akan semakin tebal pula *border* dari tabel bersangkutan. Berikut ini contoh kode yang merupakan modifikasi dari kode sebelumnya, dengan menambahkan atribut BORDER ke dalamnya.

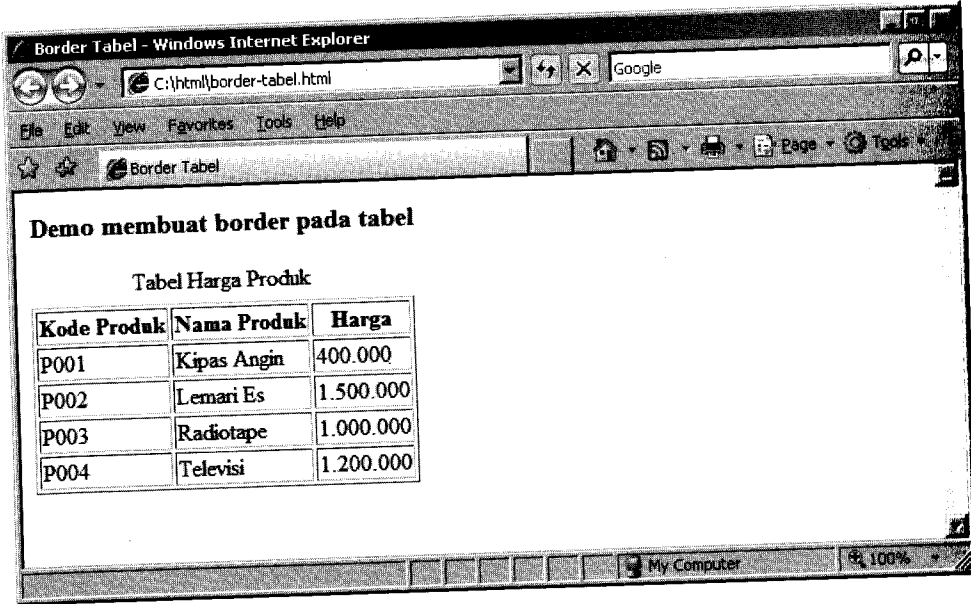
```
<html>
<head>
  <title>Border Tabel </title>
</head>
<body>

<h3>Demo membuat border pada tabel</h3>

<table border="1">
  <caption>Tabel Harga Produk</caption>
  <tr>
    <th>Kode Produk</th>
    <th>Nama Produk</th>
    <th>Harga</th>
  </tr>
  <tr>
    <td>P001</td>
    <td>Kipas Angin</td>
    <td>400.000</td>
  </tr>
  <tr>
    <td>P002</td>
    <td>Lemari Es</td>
    <td>1.500.000</td>
  </tr>
  <tr>
    <td>P003</td>
    <td>Radiotape</td>
    <td>1.000.000</td>
```

```
</tr>
<tr>
  <td>P004</td>
  <td>Televisi</td>
  <td>1.200.000</td>
</tr>
</table>

</body>
</html>
```



Untuk memahami cara kerjanya, silakan Anda lakukan perubahan nilai pada atribut BORDER dengan nilai 2, 3, atau yang lainnya, dan amati hasil yang diberikan di layar *web browser*.

Mengatur Lebar dan Tinggi Sel

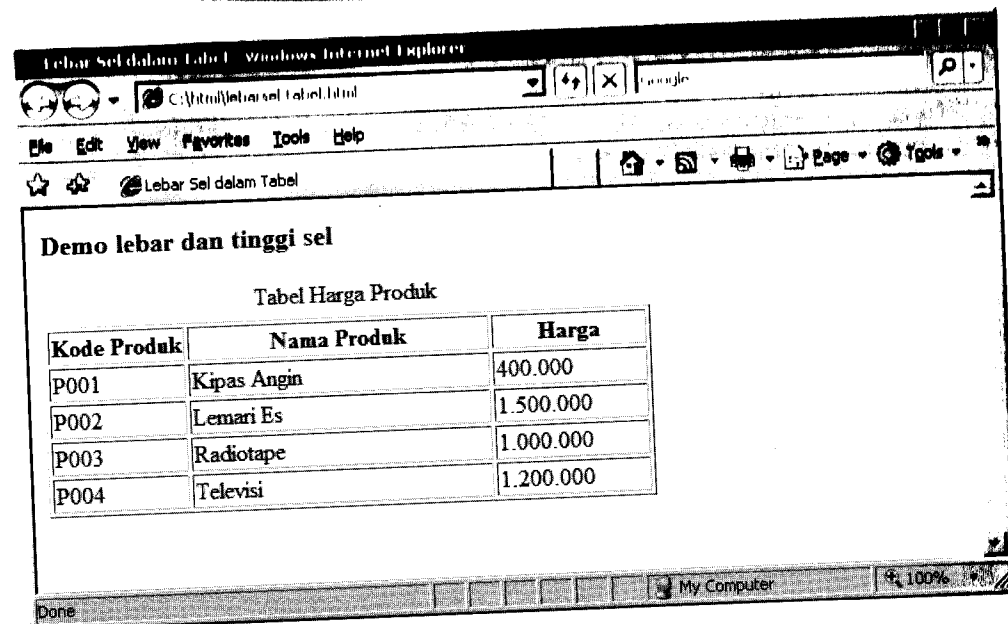
Untuk mengatur lebar dan/atau tinggi suatu sel di dalam tabel, kita hanya perlu mengisi nilai yang diinginkan ke dalam atribut WIDTH dan/atau

HEIGHT pada tag `<th>` atau `<td>`. Sebagai contoh, misalnya kita ingin melebarkan kolom kedua dan ketiga dari tabel di atas, maka kita dapat mengubah kodenya menjadi seperti berikut:

```
<html>
<head>
  <title>Lebar Sel dalam Tabel</title>
</head>
<body>
<h3>Demo lebar dan tinggi sel</h3>

<table border="1">
  <caption>Tabel Harga Produk</caption>
  <tr>
    <th>Kode Produk</th>
    <th width="200">Nama Produk</th>
    <th width="100">Harga</th>
  </tr>
  <tr>
    <td>P001</td>
    <td width="200">Kipas Angin</td>
    <td width="100">400.000</td>
  </tr>
  <tr>
    <td>P002</td>
    <td width="200">Lemari Es</td>
    <td width="100">1.500.000</td>
  </tr>
  <tr>
    <td>P003</td>
    <td width="200">Radiotape</td>
    <td width="100">1.000.000</td>
  </tr>
  <tr>
    <td>P004</td>
    <td width="200">Televisi</td>
    <td width="100">1.200.000</td>
  </tr>
</table>

</body>
</html>
```



Memberi Warna Background pada Baris dan Sel

Masing-masing baris dan sel yang terdapat di dalam tabel dapat diberi warna latar (*background*). Caranya adalah dengan memberi nilai pada atribut `BGColor` pada tag `<tr>` (untuk baris) maupun `<th>` atau `<td>` (untuk sel). Berikut ini contoh kode yang akan menunjukkan hal tersebut.

```
<html>
<head>
  <title>Warna Sel dalam Tabel</title>
</head>
<body>
<h3>Demo memberi warna pada sel</h3>

<table border="1">
  <caption>Tabel Harga Produk</caption>
  <tr bgColor="#ff99aa">
    <th>Kode Produk</th>
    <th width="200">Nama Produk</th>
    <th width="100">Harga</th>
```

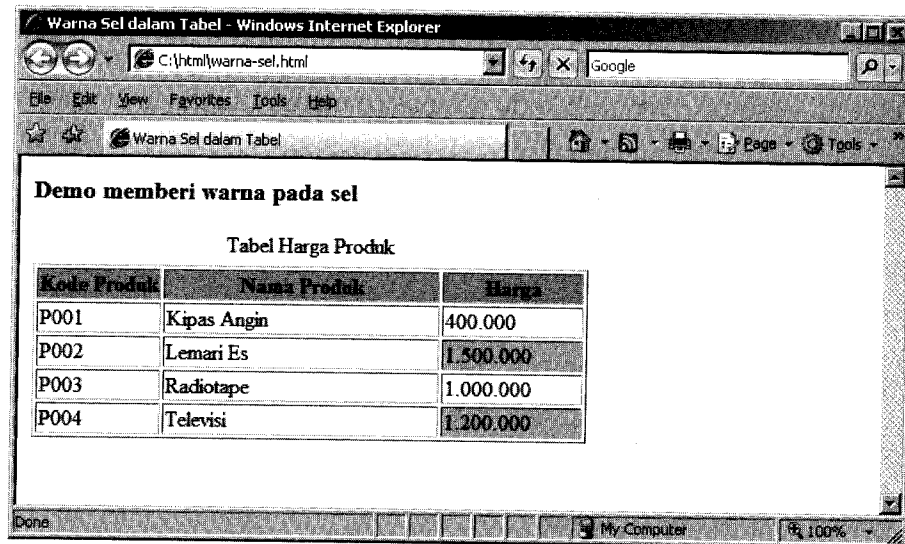


```

</td>
<tr>
<td>P001</td>
<td width="200">Kipas Angin</td>
<td width="100">400.000</td>
</tr>
<tr>
<td>P002</td>
<td width="200">Lemari Es</td>
<td width="100" bgColor="#cccccc">1.500.000</td>
</tr>
<tr>
<td>P003</td>
<td width="200">Radiotape</td>
<td width="100">1.000.000</td>
</tr>
<tr>
<td>P004</td>
<td width="200">Televisi</td>
<td width="100" bgColor="#cccccc">1.200.000</td>
</tr>
</table>

</body>
</html>

```



Memasukkan Gambar ke dalam Tabel

Dalam pembuatan dokumen HTML, kita juga diizinkan untuk memasukkan gambar ke dalam tabel. Caranya sederhana, yaitu dengan menempatkan tag `<img>` di antara tag `<td>` dan `</td>`. Contoh kode di bawah ini akan menunjukkan cara melakukan hal tersebut.

```

<html>
<head>
<title>Gambar dalam Tabel</title>
</head>
<body>
<h3>Demo memasukkan gambar ke dalam tabel</h3>

<table border="1">
<caption>Tabel Harga Produk</caption>
<tr>
<th>Kode Produk</th>
<th width="200">Nama Produk</th>
<th width="100">Harga</th>
<th width="150">Gambar</th>
</tr>
<tr>
<td>P001</td>
<td width="200">Kipas Angin</td>
<td width="100">400.000</td>
<td width="150" align="center">

</td>
</tr>
<tr>
<td>P002</td>
<td width="200">Lemari Es</td>
<td width="100">1.500.000</td>
<td width="150" align="center">

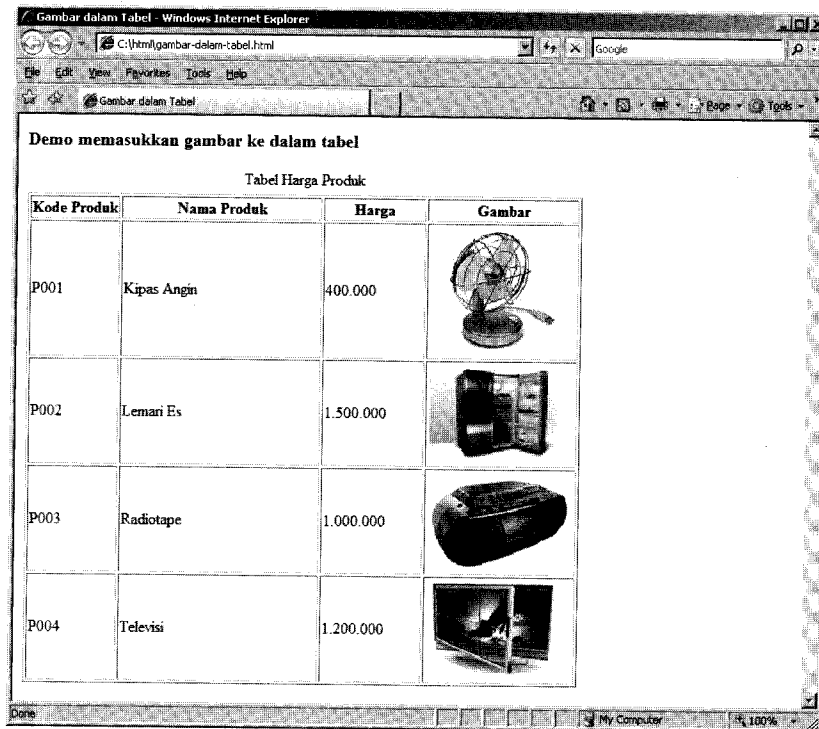
</td>
</tr>
<tr>
<td>P003</td>
<td width="200">Radiotape</td>

```

```

<td width="100">1.000.000</td>
<td width="150" align="center">
  
</td>
</tr>
<tr>
  <td>P004</td>
  <td width="200">Televisi</td>
  <td width="100">1.200.000</td>
  <td width="150" align="center">
    
  </td>
</tr>
</table>
</body>
</html>

```



Atribut CELLSPACING dan CELLPADDING

Tag <table> memiliki dua atribut yang cukup berguna dalam pembuatan tabel, yaitu CELLSPACING dan CELLPADDING. Kedua atribut ini dapat digunakan secara terpisah maupun bersama-sama.

Atribut CELLSPACING digunakan untuk mengatur jarak atau spasi antarsel (dalam satuan piksel). Dalam keadaan *default*, nilai dari atribut ini dianggap 0 (nol). Berikut ini contoh kode yang akan menunjukkan penggunaan atribut CELLSPACING.

```

<html>
<head>
  <title>Cellspacing</title>
</head>
<body>
  <h3>Demo menggunakan atribut CELLSPACING</h3>

  <table border="1" cellspacing="5">
    <caption>Tabel Harga Produk</caption>
    <tr>
      <th>Kode Produk</th>
      <th width="200">Nama Produk</th>
      <th width="100">Harga</th>
    </tr>
    <tr>
      <td>P001</td>
      <td width="200">Kipas Angin</td>
      <td width="100">400.000</td>
    </tr>
    <tr>
      <td>P002</td>
      <td width="200">Lemari Es</td>
      <td width="100">1.500.000</td>
    </tr>
    <tr>
      <td>P003</td>
      <td width="200">Radiotape</td>
      <td width="100">1.000.000</td>
    </tr>
  </table>

```

```

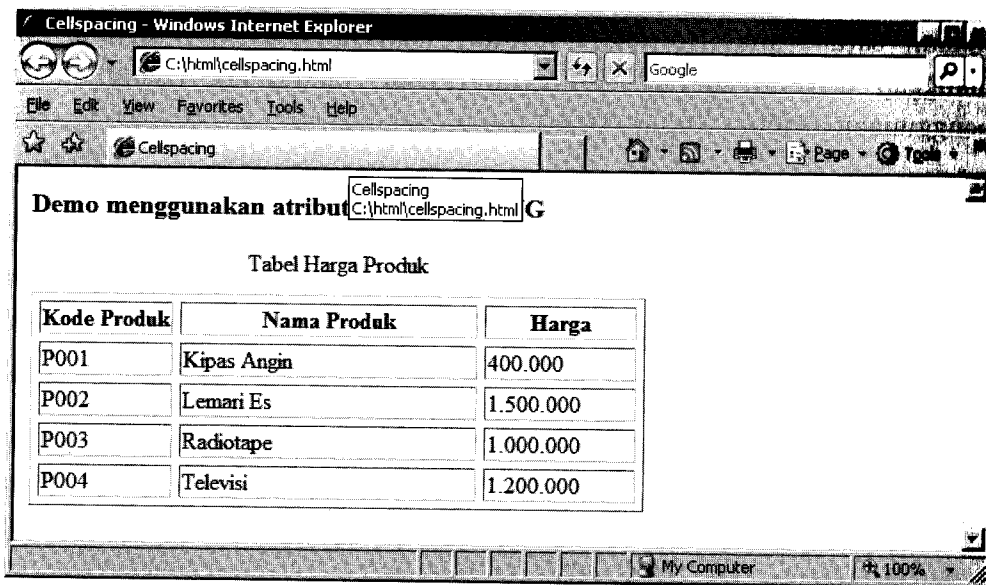
<td>P004</td>
<td width="200">Televisi</td>
<td width="100">1.200.000</td>
</tr>
</table>

```

```

</body>
</html>

```



Pada hasil di atas dapat dilihat dengan jelas bahwa jarak atau spasi antarsel tampak lebih renggang dari tabel normal. Hal ini, disebabkan oleh karena nilai atribut CELLSPACING kita isi dengan nilai 5.

Berbeda dengan CELLSPACING, atribut CELLPADDING digunakan untuk menentukan jarak antara data yang ada dalam suatu sel dengan border atau bingkai dari sel bersangkutan. Secara default, atribut ini juga bernilai 0 (nol). Contoh kode di bawah ini akan menunjukkan penggunaan atribut CELLPADDING.

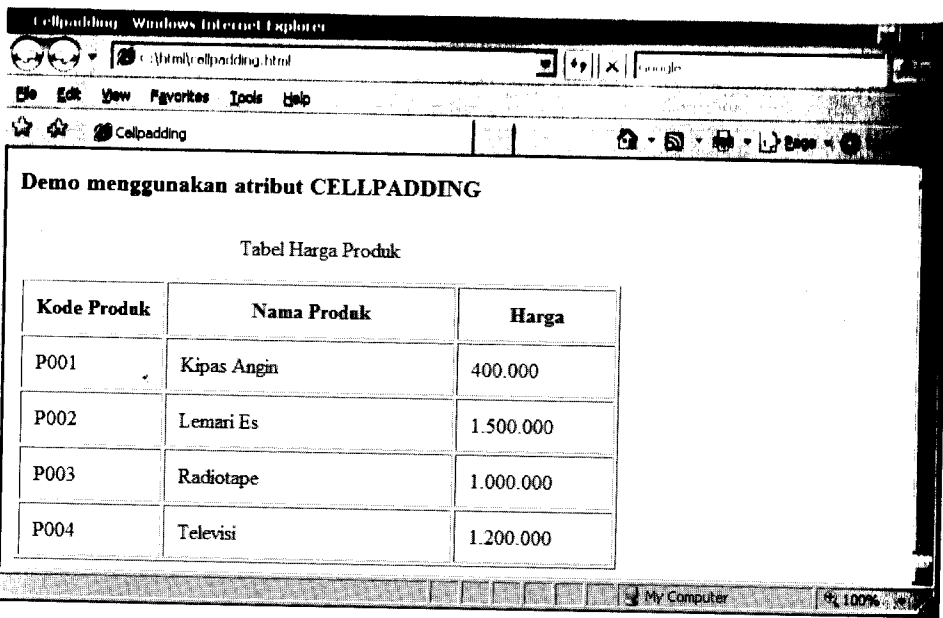
```

<html>
<head>
<title>Cellpadding</title>
</head>
<body>

<h3>Demo menggunakan atribut CELLPADDING</h3>

<table border="1" cellpadding="10">
<caption>Tabel Harga Produk</caption>
<tr>
<th>Kode Produk</th>
<th width="200">Nama Produk</th>
<th width="100">Harga</th>
</tr>
<tr>
<td>P001</td>
<td width="200">Kipas Angin</td>
<td width="100">400.000</td>
</tr>
<tr>
<td>P002</td>
<td width="200">Lemari Es</td>
<td width="100">1.500.000</td>
</tr>
<tr>
<td>P003</td>
<td width="200">Radiotape</td>
<td width="100">1.000.000</td>
</tr>
<tr>
<td>P004</td>
<td width="200">Televisi</td>
<td width="100">1.200.000</td>
</tr>
</table>
</body>
</html>

```

Atribut ROWSPAN dan COLSPAN

Dalam beberapa kasus, ada kalanya kita dituntut untuk membuat tabel yang relatif lebih rumit dibandingkan dengan tabel normal. Sel dalam tabel HTML dapat dibentangkan berdasarkan kolom maupun baris, dengan menggunakan atribut ROWSPAN dan COLSPAN, yang masing-masing merupakan atribut dari tag <th> atau <td>.

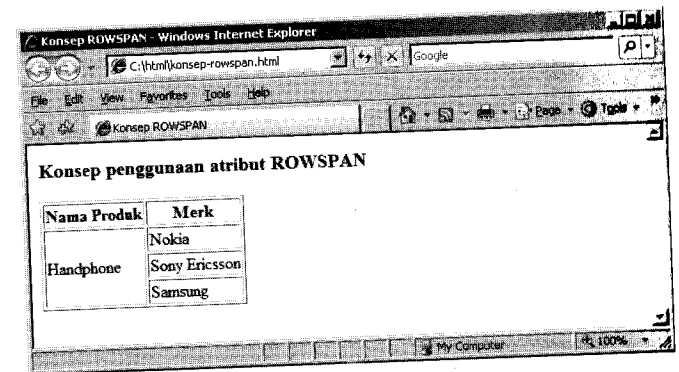
Atribut ROWSPAN digunakan untuk membentangkan sel menjadi beberapa baris. Berikut ini contoh kode yang akan menunjukkan cara kerja atribut ROWSPAN dalam tag <td>.

```
<html>
<head>
  <title>Konsep ROWSPAN</title>
</head>
<body>
  <h3>Konsep penggunaan atribut ROWSPAN</h3>
```

```
<table border="1">
  <tr>
    <th>Nama Produk</th>
    <th>Merk</th>
  </tr>
  <tr>
    <td rowspan="3">Handphone</td>
    <td>Nokia</td>
  </tr>
  <tr>
    <td>Sony Ericsson</td>
  </tr>
  <tr>
    <td>Samsung</td>
  </tr>
</table>

</body>
</html>
```

Coba Anda perhatikan baris ke-2 dan kolom ke-2 dari tabel tersebut. Baris tersebut dibagi lagi menjadi tiga baris (Nokia, Sony Ericsson, dan Samsung). Maka dari itu, sel **Handphone** harus kita bentangkan menjadi 3 baris, yaitu dengan mengisi nilai 3 ke dalam atribut ROWSPAN.



Berikut ini contoh lain yang menerapkan penggunaan atribut ROWSPAN di dalam sebuah tabel.

```
<html>
<head>
  <title>Rowspan</title>
```

```
</head>
<body>
```

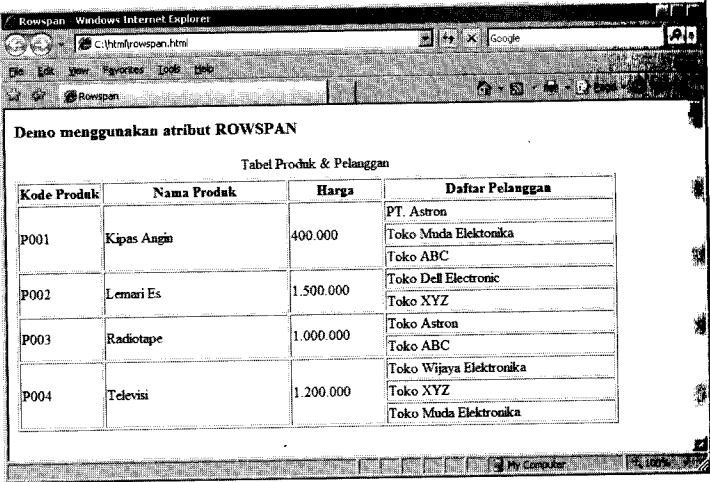
```
<h3>Demo menggunakan atribut ROWSPAN</h3>
```

```
<table border="1">
  <caption>Tabel Produk & Pelanggan</caption>
  <tr>
    <th>Kode Produk</th>
    <th width="200">Nama Produk</th>
    <th width="100">Harga</th>
    <th width="250">Daftar Pelanggan</th>
  </tr>
  <tr>
    <td rowspan="3">P001</td>
    <td width="200" rowspan="3">Kipas Angin</td>
    <td width="100" rowspan="3">400.000</td>
    <td width="250">PT. Astron</td>
  </tr>
  <tr>
    <td>Toko Muda Elektonika</td>
  </tr>
  <tr>
    <td>Toko ABC</td>
  </tr>
  <tr>
    <td rowspan="2">P002</td>
    <td width="200" rowspan="2">Lemari Es</td>
    <td width="100" rowspan="2">1.500.000</td>
    <td width="250">Toko Dell Electronic</td>
  </tr>
  <tr>
    <td>Toko XYZ</td>
  </tr>
  <tr>
    <td rowspan="2">P003</td>
    <td width="200" rowspan="2">Radiotape</td>
    <td width="100" rowspan="2">1.000.000</td>
    <td width="250">Toko Astron</td>
  </tr>
  <tr>
    <td>Toko ABC</td>
  </tr>
  <tr>
    <td rowspan="3">P004</td>
    <td width="200" rowspan="3">Televisi</td>
    <td width="100" rowspan="3">1.200.000</td>
    <td width="250">Toko Wijaya Elektronika</td>
  </tr>
  <tr>
    <td>Toko XYZ</td>
  </tr>
  <tr>
    <td>Toko Muda Elektronika</td>
  </tr>
```

```
<td rowspan="3">P004</td>
<td width="200" rowspan="3">Televisi</td>
<td width="100" rowspan="3">1.200.000</td>
<td width="250">Toko Wijaya Elektronika</td>
</tr>

<tr>
  <td>Toko XYZ</td>
</tr>
<tr>
  <td>Toko Muda Elektronika</td>
</tr>
</table>
</body>
</html>
```

Atribut COLSPAN digunakan untuk membentang sel menjadi beberapa kolom. Berikut ini contoh kode yang akan menunjukkan cara kerja atribut COLSPAN.



```
<html>
<head>
  <title>Konsep COLSPAN</title>
</head>
<body>

<h3>Konsep penggunaan atribut COLSPAN</h3>

<table border="1">
  <tr>
    <th>Nama Produk</th>
    <th colspan="3">Merk</th>
  </tr>
```

```

</tr>
<tr>
<td>Handphone</td>
<td>Nokia</td>
<td>Sony Ericsson</td>
<td>Samsung</td>
</tr>
</table>

</body>
</html>

```

Seperti yang Anda lihat pada hasil di atas, kolom kedua dari tabel tersebut dibagi lagi menjadi 3 subkolom. Untuk menangani masalah seperti ini, kita perlu membentangkan sel

Merk dengan lebar menjadi 3 kolom, yaitu dengan cara mengisi nilai 3 ke dalam atribut COLSPAN.

Adapun contoh implementasi yang lebih kompleks dapat dilihat pada kode di bawah ini.

```

<html>
<head>
<title>Colspan</title>
</head>
<body>

<h3>Demo menggunakan atribut COLSPAN</h3>

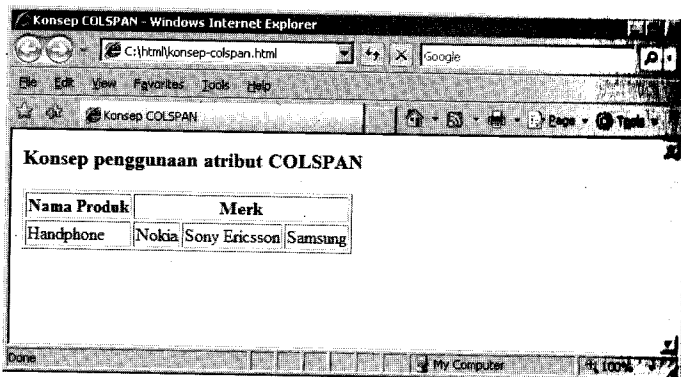
<table border="1">
<caption>Tabel Rekapitulasi Penjualan Produk</caption>
<tr>
<th rowspan="2" width="80">Tahun</th>

```

```

<th rowspan="2" width="150">Produk</th>
<th colspan="12">Bulan</th>
</tr>
<tr>
<th width="50">Jan</th>
<th width="50">Feb</th>
<th width="50">Mar</th>
<th width="50">Apr</th>
<th width="50">Mei</th>
<th width="50">Jun</th>
<th width="50">Jul</th>
<th width="50">Agu</th>
<th width="50">Sep</th>
<th width="50">Okt</th>
<th width="50">Nov</th>
<th width="50">Des</th>
</tr>
<tr>
<td align="center" rowspan="4">2007</td>
<td align="left">Kipas Angin</td>
<td align="center">23</td>
<td align="center">30</td>
<td align="center">15</td>
<td align="center">40</td>
<td align="center">33</td>
<td align="center">11</td>
<td align="center">60</td>
<td align="center">53</td>
<td align="center">90</td>
<td align="center">24</td>
<td align="center">12</td>
<td align="center">56</td>
</tr>
<tr>
<td align="left">Lemari Es</td>
<td align="center">31</td>
<td align="center">22</td>
<td align="center">24</td>
<td align="center">46</td>
<td align="center">39</td>
<td align="center">29</td>
<td align="center">40</td>
<td align="center">64</td>
<td align="center">85</td>

```




```

<td align="center">29</td>
<td align="center">20</td>
<td align="center">70</td>
</tr>
<tr>
<td align="left">Radiotape</td>
<td align="center">20</td>
<td align="center">13</td>
<td align="center">19</td>
<td align="center">38</td>
<td align="center">42</td>
<td align="center">29</td>
<td align="center">33</td>
<td align="center">58</td>
<td align="center">99</td>
<td align="center">35</td>
<td align="center">32</td>
<td align="center">50</td>
</tr>
<tr>
<td align="left">Televisi</td>
<td align="center">26</td>
<td align="center">43</td>
<td align="center">45</td>
<td align="center">20</td>
<td align="center">21</td>
<td align="center">34</td>
<td align="center">64</td>
<td align="center">62</td>
<td align="center">88</td>
<td align="center">31</td>
<td align="center">25</td>
<td align="center">63</td>
</tr>
<tr>
<td align="center" rowspan="4">2008</td>
<td align="left">Kipas Angin</td>
<td align="center">23</td>
<td align="center">30</td>
<td align="center">15</td>
<td align="center">40</td>
<td align="center">33</td>
<td align="center">11</td>
<td align="center">60</td>

```

```

<td align="center">53</td>
<td align="center">90</td>
<td align="center">24</td>
<td align="center">12</td>
<td align="center">56</td>
</tr>
<tr>
<td align="left">Lemari Es</td>
<td align="center">13</td>
<td align="center">40</td>
<td align="center">17</td>
<td align="center">55</td>
<td align="center">37</td>
<td align="center">18</td>
<td align="center">65</td>
<td align="center">57</td>
<td align="center">91</td>
<td align="center">27</td>
<td align="center">18</td>
<td align="center">59</td>
</tr>
<tr>
<td align="left">Radiotape</td>
<td align="center">33</td>
<td align="center">20</td>
<td align="center">25</td>
<td align="center">50</td>
<td align="center">43</td>
<td align="center">21</td>
<td align="center">50</td>
<td align="center">63</td>
<td align="center">70</td>
<td align="center">44</td>
<td align="center">32</td>
<td align="center">16</td>
</tr>
<tr>
<td align="left">Televisi</td>
<td align="center">42</td>
<td align="center">56</td>
<td align="center">34</td>
<td align="center">36</td>
<td align="center">37</td>
<td align="center">17</td>

```

```

<td align="center">66</td>
<td align="center">55</td>
<td align="center">92</td>
<td align="center">28</td>
<td align="center">34</td>
<td align="center">59</td>
</tr>
<tr>
<td align="center" rowspan="4">2009</td>
<td align="left">Kipas Angin</td>
<td align="center">23</td>
<td align="center">30</td>
<td align="center">15</td>
<td align="center">40</td>
<td align="center">33</td>
<td align="center">11</td>
<td align="center" colspan="6">
<i>-- masih berlangsung --</i></td>
</tr>
<tr>
<td align="left">Lemari Es</td>
<td align="center">39</td>
<td align="center">40</td>
<td align="center">35</td>
<td align="center">20</td>
<td align="center">53</td>
<td align="center">41</td>
<td align="center" colspan="6">
<i>-- masih berlangsung --</i></td>
</tr>
<tr>
<td align="left">Radiotape</td>
<td align="center">38</td>
<td align="center">39</td>
<td align="center">40</td>
<td align="center">43</td>
<td align="center">63</td>
<td align="center">60</td>
<td align="center" colspan="6">
<i>-- masih berlangsung --</i></td>
</tr>
<tr>
<td align="left">Televisi</td>
<td align="center">42</td>

```

```

<td align="center">33</td>
<td align="center">25</td>
<td align="center">40</td>
<td align="center">35</td>
<td align="center">38</td>
<td align="center" colspan="6">
<i>-- masih berlangsung --</i></td>
</tr>
</table>

</body>
</html>

```

Colspan - Windows Internet Explorer

C:\html\colspan.html

File Edit View Favorites Tools Help

Colspan

Demo menggunakan atribut COLSPAN

Tabel Rekapitulasi Penjualan Produk

Tahun	Produk	Bulan											
		Jan	Feb	Mar	Apr	Mei	Jun	Jul	Agus	Sep	Okt	Nov	Des
2007	Kipas Angin	23	30	15	40	33	11	60	53	90	24	12	56
	Lemari Es	31	22	24	46	39	29	40	64	85	29	20	70
	Radiotape	20	13	19	38	42	29	33	58	99	35	32	50
	Televisi	26	43	45	20	21	34	64	62	88	31	25	63
2008	Kipas Angin	23	30	15	40	33	11	60	53	90	24	12	56
	Lemari Es	13	40	17	55	37	18	65	57	91	27	18	59
	Radiotape	33	20	25	50	43	21	50	63	70	44	32	16
	Televisi	42	56	34	36	37	17	66	55	92	28	34	59
2009	Kipas Angin	23	30	15	40	33	11	-- masih berlangsung --					
	Lemari Es	39	40	35	20	53	41	-- masih berlangsung --					
	Radiotape	38	39	40	43	63	60	-- masih berlangsung --					
	Televisi	42	33	25	40	35	38	-- masih berlangsung --					

My Computer 100%



Referensi MySQL

Pengenalan *Database*

Bagian ini akan menjelaskan dasar-dasar *database*, yang meliputi: pengertian *database*, pengertian tabel, pengertian SQL, dan pengenalan komponen-komponen penting dalam SQL.

Dasar-dasar Database

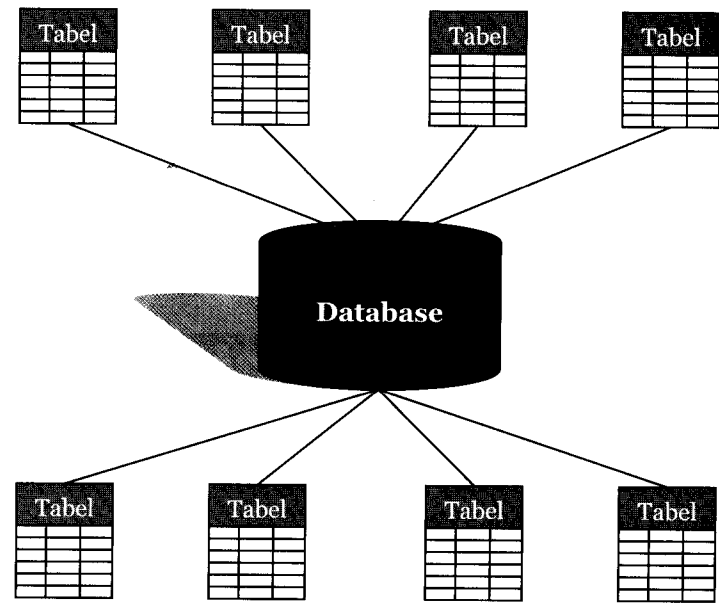
Dalam mempelajari SQL, kita selalu berinteraksi dengan *database*. Maka dari itu, sebelum kita membahas tentang SQL itu sendiri, kita perlu

mengkaji sekilas mengenai pengertian *database* beserta elemen-elemen yang terdapat di dalamnya.

Apa itu Database?

Secara umum, *database* dapat didefinisikan sebagai kumpulan **tabel** (walaupun lebih tepat dikatakan kumpulan objek karena yang terkandung di dalam *database* sebenarnya bukan hanya tabel, melainkan indeks, *view*, *constraint*, *trigger*, dan sebagainya). Dalam sistem *database* relasional atau RDBMS (*Relational Database Management System*), tabel-tabel tersebut harus saling berelasi melalui kolom-kolom yang ada berdasarkan aturan-aturan tertentu.

Gambar di bawah ini akan mengilustrasikan suatu *database*.



Disadari atau tidak, *database* sebenarnya selalu diimplementasikan dalam kehidupan kita sehari-hari, setiap saat. Sebagai contoh, ketika kita mengambil uang dari mesin ATM, kita sedang menggunakan *database*.

Contoh lainnya, ketika kita melakukan pencarian informasi di internet melalui Google atau mesin pencari lainnya, kita juga sebenarnya sedang menggunakan *database*. Dengan menggunakan *database*, terjadi adanya sentralisasi data sehingga mudah dalam penyimpanan, pengaturan, dan pencariannya.

Tabel

Seperti yang telah disebutkan sebelumnya bahwa *database* secara mudah dapat dikatakan sebagai kumpulan tabel. Namun, apa yang dimaksud tabel itu sendiri? Tabel adalah kumpulan data yang ditampilkan secara tabular dan dipisahkan berdasarkan kolom-kolom tertentu. Terdapat tiga komponen penting di dalam suatu tabel yang perlu Anda ketahui kaitannya dengan *database*.

- Kolom dan Tipe Data
- Baris
- Sel

Contoh tabel dapat dilihat di bawah ini.

NIP	Nama Lengkap	Alamat	Kota	Telp
103.121.001	Ardi Susanto	Jl. Kenari 212	Jakarta	021-80922333
103.121.002	Aura Kasih	Jl. Murai Batu 444	Jakarta	021-91922444
103.121.003	Benu Bolue	Jl. Cucak Rawa 13	Bandung	022-73333333
103.121.004	Cindy Claudia	Jl. Gagak 599	Bandung	022-20166666
103.121.005	Zaliyanti Olivia	Jl. Anis Kembang 988	Semarang	024-54911188

Setiap tabel di dalam *database* harus diidentifikasi dengan nama yang unik.

Kolom dan Tipe Data

Tabel terbuat atas kolom-kolom yang berisi bagian atau potongan informasi di dalam tabel. Tipe data dari kolom-kolom yang terdapat pada sebuah tabel dapat berbeda dan dapat kita tentukan sendiri.

Dalam dunia *database*, kolom sering disebut sebagai *field*.

Baris

Data dalam suatu tabel disimpan di dalam baris, yang dibentangkan secara horizontal berdasarkan kolom-kolom yang telah didefinisikan.

Dalam dunia *database*, baris sering disebut dengan istilah *record*.

Primary Key

Primary key adalah *field* kunci dalam suatu tabel, yang berfungsi sebagai pembeda antara *record* yang satu dengan yang lain. *Primary key* akan memastikan bahwa dalam tabel bersangkutan tidak akan ada *record* yang isinya sama persis.

Foreign Key

Foreign key adalah *field* yang dijadikan sebagai kunci tamu dalam suatu tabel. Isi dari *field* ini akan selalu mengacu pada isi *field* yang terdapat pada tabel lain. Dengan konsep inilah, dua atau beberapa tabel dapat saling direlasikan.

Apa Itu SQL?

SQL (biasa dibaca *sequel*) adalah kependekan dari *Structured Query Language*. SQL merupakan bahasa yang dirancang untuk berkomunikasi

dengan *database*. Tidak seperti bahasa pemrograman (seperti C, Visual Basic, dan sebagainya), SQL hanya memiliki beberapa kata kunci saja. Tujuan dirancangnya SQL adalah untuk mengefisienkan dan menyederhanakan kita dalam membaca atau menulis data dari/ke dalam *database*.

Selain mendukung SQL standar, setiap DBMS pada umumnya menambahkan sendiri fitur-fitur khusus dengan menyediakan statemen atau perintah-perintah tambahan.

Tiga Elemen Penting dalam SQL

Tiga elemen penting yang perlu Anda ketahui dalam mempelajari SQL adalah sebagai berikut:

- *Data Definition Language* (DDL), yaitu statemen-statemen yang berhubungan dengan pembuatan objek (misalnya tabel) dan pengelolaan strukturnya.
- *Data Manipulation Language* (DML), yaitu statemen-statemen yang berhubungan dengan pemanipulasian data di dalam tabel.
- *Data Control Language* (DCL), yaitu statemen-statemen kontrol seperti COMMIT dan ROLLBACK.

Mengenal Statemen SELECT

Pengambilan data dari dalam *database* dilakukan dengan menggunakan statemen SELECT. Pada saat kita bekerja dengan *database*, kita akan banyak menggunakan statemen ini. Bentuk umum penggunaannya adalah sebagai berikut:

```
select field1, field2, ..., fieldN from tabel
```

Untuk mengakses semua *field*, kita dapat menggunakan tanda asterisk (*).

Membuat dan Memanipulasi Struktur Tabel

Tabel merupakan komponen utama atau inti dalam suatu *database*. Pada bagian ini kita akan mempelajari cara pembuatan dan manipulasi struktur tabel.

Membuat Tabel

Tabel di dalam *database* dibuat menggunakan statemen `CREATE TABLE`. Bentuk umumnya adalah sebagai berikut:

```
create table nama (  
    field1 tipe,  
    field2 tipe,  
    ...  
);
```

Contoh:

```
create table produk (  
    id char(4),  
    nama varchar(30),  
    harga numeric(8,2),  
    deskripsi varchar(255)  
);
```

Sintaks SQL di atas akan membuat tabel `PRODUK` di dalam *database* dengan kolom `ID`, `NAMA`, `HARGA`, dan `DESKRIPSI`.

Null / Not Null

Field dalam *database* dapat berisi nilai `NULL` (nilai kosong). Jika Anda ingin memastikan bahwa suatu *field* harus diisi, maka tambahkan klausa `NOT NULL` pada saat mendefinisikan *field*. Beberapa DBMS mewajibkan kita untuk membuat *field* `NOT NULL` jika kita ingin menjadikan *field* tersebut sebagai *primary key*.

Contoh:

```
create table produk (  
    id char(4) not null,  
    nama varchar(30),  
    harga numeric(8,2),  
    deskripsi varchar(255)  
);
```

Dengan definisi seperti di atas, *field* `ID` tidak boleh berisi nilai `NULL`.

Menambah Constraint

Constraint (batasan/aturan tertentu) dapat didefinisikan pada saat pembuatan tabel, bisa juga menyusul, yaitu dengan penggunaan statemen `ALTER TABLE`. Bentuk umumnya adalah sebagai berikut:

Primary key:

```
constraint nama primary key (NamaField)
```

Foreign key:

```
constraint nama foreign key (NamaField) references TabelLain  
(NamaField)
```

Contoh pembuatan *constraint* untuk *primary key* pada tabel `PRODUK`:

```
create table produk
(
  id char(4) not null,
  nama varchar(50),
  harga numeric(8,2),
  deskripsi varchar(255),
  constraint pk_produk primary key(id)
);
```

Jika penambahannya dibuat setelah pembuatan tabel, maka kita dapat menggunakan sintaks berikut:

```
alter table produk
add constraint pk_produk primary key(id);
```

Menambah Kolom Tabel

Tabel yang telah dibuat dapat pula ditambahkan *field-field* baru. Sebagai contoh, kita ingin menambahkan *field* ID_VENDOR pada tabel PRODUK. Untuk melakukan perubahan terhadap struktur tabel, kita harus menggunakan statemen ALTER TABLE.

Berikut ini adalah bentuk umum yang digunakan untuk menambah kolom ke dalam suatu tabel:

```
alter table NamaTable
add column Field Tipe;
```

Contoh:

```
alter table produk
add column id_vendor char(3) not null;
```

Dalam kasus ini, *field* ID_VENDOR dapat digunakan sebagai *foreign key* yang isinya mengacu pada tabel lain, misalnya VENDOR.

Mengubah Kolom Tabel

Kolom atau *field* dari suatu tabel yang telah didefinisikan sebelumnya masih dapat diubah sesuai kebutuhan. Berikut ini contoh yang akan menunjukkan hal tersebut.

```
alter table produk
modify column id char(6);
```

Pada contoh di atas kita mengubah lebar *field* dari 4 menjadi 6 karakter.

Menghapus Kolom Tabel

Selain penambahan dan pengubahan, kita juga dapat menghapus satu atau beberapa *field* dari suatu tabel. Contoh sintaks SQL yang dapat menunjukkan hal tersebut adalah sebagai berikut:

```
alter table produk
drop column id_vendor;
```

Melalui sintaks tersebut, tabel ID_VENDOR pada tabel PRODUK akan terhapus.

Menghapus Tabel

Tabel yang sudah tidak diperlukan dalam suatu *database* atau tabel yang ingin dibangun ulang dari awal, dapat dihapus dengan menggunakan statemen DROP TABLE. Bentuk umum penggunaan statemen ini adalah sebagai berikut:

```
drop table nama
```

Contoh:

```
drop table produk
```

Perintah SQL di atas akan menghapus tabel `PRODUK` beserta isinya (jika ada) secara permanen dari dalam *database*.

Hal yang perlu diperhatikan pada saat menghapus sebuah tabel adalah masalah *constraint*. Pastikan tabel yang akan dihapus tidak berisi data yang sedang diacu oleh tabel lain.

Latihan Membuat Tabel

Untuk keperluan studi kasus dalam modul ini, silakan Anda tuliskan sintaks SQL berikut ini.

```
/* Tabel Penerbit */
create table penerbit (
    kode_penerbit char(2) not null,
    nama_penerbit varchar(50),
    constraint pk_penerbit primary key(kode_penerbit)
);
```

```
/* Tabel Siswa */
create table siswa (
    nim char(6) not null,
    nama varchar(30),
    tgl_lahir date,
    gender char(1),
    alamat varchar(50),
    kota varchar(20),
    telepon char(15),
    constraint pk_siswa primary key(nim)
);
```

```
/* Tabel Buku */
create table buku (
    kode_buku char(4) not null,
    judul varchar(50),
    pengarang varchar(30),
    kode_penerbit char(2),
    jumlah_buku integer,
```

```
    constraint pk_buku primary key(kode_buku),
    constraint fk_buku_penerbit foreign key(kode_penerbit)
        references penerbit(kode_penerbit)
);
```

```
/* Tabel Peminjaman Buku */
create table peminjaman (
    tgl_pinjam date not null,
    nim char(6) not null,
    kode_buku char(4) not null,
    tgl_batas_kembali date,
    tgl_kembali date,
    constraint pk_peminjaman primary key(tgl_pinjam,
        nim, kode_buku),
    constraint fk_peminjaman_siswa foreign key(nim)
        references siswa(nim),
    constraint fk_peminjaman_buku foreign key(kode_buku)
        references buku(kode_buku)
);
```

Memanipulasi Data

Bagian ini akan mempelajari cara memasukkan data ke dalam suatu tabel, mengubah data, dan menghapus data berdasarkan kriteria atau kondisi tertentu.

Memasukkan Data Baru

Setelah melakukan pendefinisian tabel-tabel yang diperlukan dalam suatu *database*, langkah selanjutnya adalah memasukkan data-data ke dalam tabel tersebut. Untuk melakukan hal ini, perintah SQL yang perlu digunakan adalah statemen `INSERT`, yang memiliki bentuk umum seperti berikut:

```
insert into NamaTabel (field1, field2, ...) values (nilai1, nilai2, ...)
```


Jika semua *field* akan diisi, maka nama *field* tidak perlu dituliskan, seperti yang terlihat pada bentuk umum berikut:

```
insert into NamaTabel values(nilai1, nilai2, ...)
```

Berikut ini perintah yang dapat digunakan untuk memasukkan contoh data baru ke dalam tabel PRODUK.

```
insert into produk(id, nama, harga, deskripsi)
values('EL-001', 'Lemari ES', 1500000, null);
insert into produk(id, nama, harga, deskripsi)
values('EL-002', 'Televisi', 1300000, null);
insert into produk(id, nama, harga, deskripsi)
values('EL-003', 'Radiotape', 750000, null);
insert into produk(id, nama, harga, deskripsi)
values('EL-004', 'Kipas Angin', 250000, null);
insert into produk(id, nama, harga, deskripsi)
values('EL-005', 'Kamera Digital', 1250000, null);
insert into produk(id, nama, harga, deskripsi)
values('EL-006', 'Kompor Listrik', 450000, null);
insert into produk(id, nama, harga, deskripsi)
values('EL-007', 'Rice Cooker', 300000, null);
```

atau dapat juga disederhanakan menjadi seperti berikut:

```
insert into produk values('EL-001', 'Lemari ES',
1500000, null);
insert into produk values('EL-002', 'Televisi',
1300000, null);
insert into produk values('EL-003', 'Radiotape',
750000, null);
insert into produk values('EL-004', 'Kipas Angin',
250000, null);
insert into produk values('EL-005', 'Kamera Digital',
1250000, null);
insert into produk values('EL-006', 'Kompor Listrik',
450000, null);
insert into produk values('EL-007', 'Rice Cooker',
300000, null);
```

Perintah di atas akan membuat 7 *record* ke dalam tabel PRODUK, sehingga sekarang datanya akan tampak seperti di bawah ini.

ID	NAMA	HARGA	DESKRIPSI
EL-001	Lemari ES	1500000,0	(NULL)
EL-002	Televisi	1300000,0	(NULL)
EL-003	Radiotape	750000,0	(NULL)
EL-004	Kipas Angin	250000,0	(NULL)
EL-005	Kamera Digital	1250000,0	(NULL)
EL-006	Kompor Listrik	450000,0	(NULL)
EL-007	Rice Cooker	300000,0	(NULL)

Mengubah Data

Perubahan data di dalam suatu tabel dapat dilakukan dengan menggunakan statemen UPDATE, yang memiliki bentuk umum penggunaan seperti berikut:

```
update NamaTabel
set field1 = nilai1, field2 = nilai2, ...
where kondisi
```

Contoh:

```
update produk
set nama = 'Kulkas'
where id = 'EL-001';
```

Sintaks di atas digunakan untuk mengubah nama produk yang memiliki ID 'EL-001' dengan nama baru, yaitu 'Kulkas', sehingga datanya menjadi seperti di bawah ini.

id	nama	harga	deskripsi
EL-001	Kulkas	1500000,0	(NULL)
EL-002	Televisi	1300000,0	(NULL)
EL-003	Radiotape	750000,0	(NULL)
EL-004	Kipas Angin	250000,0	(NULL)
EL-005	Kamera Digital	1250000,0	(NULL)
EL-006	Kompas Listrik	450000,0	(NULL)
EL-007	Rice Cooker	300000,0	(NULL)

Hal yang perlu diingat dalam penggunaan statemen UPDATE adalah pendefinisian kondisi. Kondisi tersebut berfungsi untuk menunjukkan *record-record* mana saja yang akan diperbaharui.

Menghapus Data

Penghapusan data dalam suatu tabel dapat dilakukan dengan menggunakan statemen DELETE. Bentuk umumnya adalah sebagai berikut:

```
delete from NamaTabel where kondisi
```

Contoh:

```
delete from produk where id = 'EL-007'
```

Sintaks di atas akan menghapus *record* yang memiliki ID dengan nilai 'EL-007', sehingga data akan tampak seperti berikut ini.

id	nama	harga	deskripsi
EL-001	Kulkas	1500000,0	(NULL)
EL-002	Televisi	1300000,0	(NULL)
EL-003	Radiotape	750000,0	(NULL)
EL-004	Kipas Angin	250000,0	(NULL)
EL-005	Kamera Digital	1250000,0	(NULL)
EL-006	Kompas Listrik	450000,0	(NULL)

Jika kondisi tidak disertakan dalam perintah tersebut, maka seluruh *record* dalam tabel PRODUK akan terhapus. Perintah lain yang dapat digunakan untuk menghapus seluruh *record* dalam tabel adalah TRUNCATE.

Contoh:

```
truncate produk;
```

Latihan Mengisi Data ke dalam Tabel

Berikut ini contoh sintaks yang dapat digunakan untuk memasukkan data baru ke dalam tabel-tabel yang telah kita buat sebelumnya.

```
/* insert data ke tabel penerbit */
insert into penerbit values('01', 'Elex');
insert into penerbit values('02', 'Kompas');
insert into penerbit values('03', 'Andi');
insert into penerbit values('04', 'Informatika');

/* insert data ke tabel buku */
insert into buku values('K001', 'Menguasai ORACLE',
    'Budi Raharjo', '04', 10);
insert into buku values('K002', 'Pemrograman Delphi',
    'Antoni Pranata', '03', 5);
insert into buku values('K003', 'Algoritma & Pemrograman',
    'Rinaldi', '04', 15);
insert into buku values('K004', 'Pemrograman PHP',
    'Betha S', '04', 8);
insert into buku values('K005', 'Map Server',
    'Ruslan N', '04', 2);
insert into buku values('K006', '36 Mahir Komputer',
    'Santoso', '01', 3);
insert into buku values('K007', 'Mudah Belajar Java',
    'Budi Raharjo', '04', 15);
insert into buku values('K008', 'MS Access',
    'Imam Heryanto', '04', 20);

/* insert data ke tabel siswa */
insert into siswa values('DP-001', 'Jajang L. R.',
    '1986-12-10', 'L', 'Jl. Kembar Baru Tenggara 4');
```

```

'Bandung', '022-2004477');
insert into siswa values('DP-002', 'Oma Hermawan',
'1986-07-12', 'L', 'Jl. Kacer 23',
'Bandung', '022-7662088');
insert into siswa values('DP-003', 'Alit Hasan',
'1986-01-01', 'L', 'Jl. Kacer 24',
'Bandung', '022-7664211');
insert into siswa values('WP-001', 'Eca Ane',
'1987-11-13', 'P', 'Jl. Bidakara 45',
'Bandung', '022-2021100');
insert into siswa values('WP-002', 'Asyanto',
'1985-10-07', 'L', 'Jl. Kembar Timur 7',
'Bandung', '022-4057732');

/* insert data ke tabel peminjaman */
insert into peminjaman values('2009-01-02', 'DP-003',
'K002', '2009-01-05', '2009-01-05');
insert into peminjaman values('2009-01-02', 'WP-002',
'K001', '2009-01-05', '2009-01-08');
insert into peminjaman values('2009-01-02', 'WP-002',
'K004', '2009-01-10', '2009-01-10');
insert into peminjaman values('2009-01-03', 'DP-001',
'K002', '2009-01-06', '2009-01-07');
insert into peminjaman values('2009-01-03', 'WP-001',
'K001', '2009-01-06', '2009-01-10');

```

Seleksi Data

Seleksi data adalah proses menampilkan data yang tersimpan dalam tabel-tabel di dalam *database*. Bagian ini akan mempelajari teknik pengambilan data, baik dari satu tabel maupun dari beberapa tabel yang memiliki relasi tertentu.

Menampilkan Field yang Dipilih

Data yang tersimpan dalam suatu *database* dapat ditampilkan menggunakan statemen **SELECT**. Pada kasus-kasus pemrograman tertentu,

terkadang kita hanya perlu mengambil satu atau beberapa *field* saja. Hal ini tentu disesuaikan dengan kebutuhan program atau aplikasi yang sedang kita kembangkan. Bentuk umum untuk menampilkan data dari *field-field* yang dipilih adalah sebagai berikut:

```
select field1, field2, ..., fieldN from NamaTabel
```

Contoh:

```
select id, nama from produk
```

Sintaks di atas berguna untuk menampilkan *field* ID dan NAMA yang terdapat pada tabel **PRODUK**. Hasil yang akan diperoleh jika perintah tersebut dieksekusi adalah sebagai berikut:

id	nama
EL-001	Kulkas
EL-002	Televisi
EL-003	Radiotape
EL-004	Kipas Angin
EL-005	Kamera Digital
EL-006	Kompas Listrik

Field-field yang ditampilkan dapat dipilih dari satu atau beberapa tabel. Untuk sementara ini, Anda hanya perlu tahu cara melakukan hal tersebut dari satu tabel saja. Untuk *field* yang diambil dari beberapa tabel akan dibahas pada bagian lain dari modul ini.

Menampilkan Semua Field

Untuk menampilkan semua *field* yang ada dalam suatu tabel, Anda dapat menggunakan cara di atas, yaitu dengan menyebutkan satu per satu semua *field* dari tabel bersangkutan.

```
select id, nama, harga, deskripsi from produk
```

Meskipun demikian, terdapat cara lain yang lebih sederhana, yaitu dengan menggunakan tanda *asterisk* (*). Tanda ini berfungsi untuk mewakili seluruh *field* yang terdapat pada tabel yang dipilih. Dengan demikian, sintaks di atas dapat disederhanakan menjadi seperti berikut:

```
select * from produk
```

Hasil yang akan diberikan:

id	nama	harga	deskripsi
EL-001	Kulkas	1500000,0	(NULL)
EL-002	Televisi	1300000,0	(NULL)
EL-003	Radiotape	750000,0	(NULL)
EL-004	Kipas Angin	250000,0	(NULL)
EL-005	Kamera Digital	1250000,0	(NULL)
EL-006	Kompas Listrik	450000,0	(NULL)

Menggunakan Alias

Dalam melakukan seleksi data, ada kalanya kita perlu menggunakan alias untuk mempersingkat nama tabel. Alias sebenarnya banyak digunakan pada saat kita sudah mulai memilih *field* dari beberapa tabel. Namun, di sini kita hanya akan mengambil contoh dari satu tabel saja. Perhatikan kembali sintaks berikut:

```
select id, nama, harga, deskripsi from produk
```

Perintah di atas sebenarnya sama saja bila ditulis seperti ini:

```
select
  produk.id, produk.nama,
  produk.harga, produk.deskripsi
from produk
```

Karena hal tersebut terlalu menyita waktu, maka untuk menyederhanakan penulisannya, kita dapat menggunakan alias. Dengan demikian, sintaks di atas dapat juga ditulis seperti berikut:

```
select
  p.id, p.nama, p.harga, p.deskripsi
from produk p
```

Pada contoh di atas, kita menggunakan alias **p** sebagai nama lain dari tabel PRODUK.

Memfilter Tampilan Data

Proses penyaringan (*filtering*) data adalah menampilkan data yang memenuhi kriteria-kriteria tertentu yang kita definisikan. Kriteria ini sering disebut dengan kondisi. Dan dalam SQL, kondisi harus didefinisikan menggunakan klausa **WHERE**. Dengan demikian, bentuk umum untuk proses penyaringan data dapat ditulis seperti berikut:

```
select field1, field2, ..., fieldN
from NamaTabel1, NamaTabel2, ..., NamaTabelN
where kondisi
```

Contoh:

```
select
  id, nama, harga
from produk
where harga < 1000000
```

Kita juga dapat menyertakan kondisi yang lebih dari satu dengan relasi logika **OR** atau **AND**.

```
select field1, field2, ..., fieldN
from NamaTabel1, NamaTabel2, ..., NamaTabelN
where kondisi1 [and | or]
      kondisi2 [and | or]
      kondisi3 [and | or]
...
```

Contoh:

```
select id, nama, harga from produk
where harga <= 1000000 and harga >= 500000
```

Perintah tersebut akan menampilkan data yang memiliki harga pada rentang 500000 sampai 1000000.

Untuk data-data yang bertipe karakter, kita dapat menggunakan operator LIKE, seperti yang tampak pada contoh sintaks berikut:

```
select id, nama, harga from produk
where nama like '%d%';
```

Mengurutkan Tampilan Data

Untuk keperluan pengurutan data yang akan ditampilkan, kita perlu menyertakan klausa ORDER BY pada akhir statemen SELECT yang kita tulis. Pengurutan data dapat dilakukan secara *ascending* maupun *descending*, tergantung kebutuhan aplikasi. Secara *default*, proses pengurutan akan dilakukan secara *ascending* (menaik). Pengurutan secara *descending* (menurun) dilakukan dengan membubuhkan klausa DESC di akhir penggunaan ORDER BY.

Bentuk umum:

```
[statemen SELECT] ORDER BY field1, field2, ... [ASC | DESC]
```

Contoh:

```
select id, nama, harga from produk
order by nama asc, harga desc
```

Hasil yang akan diberikan:

id	nama	harga
EL-005	Kamera Digital	1250000,0
EL-004	Kipas Angin	250000,0
EL-006	Kompor Listrik	450000,0
EL-001	Kulkas	1500000,0
EL-003	Radiotape	750000,0
EL-002	Televisi	1300000,0

Join

Seperti yang telah disebutkan sebelumnya bahwa seleksi data dapat dilakukan dari beberapa tabel yang memiliki relasi satu sama lain. Relasi tersebut dibuat menggunakan *field-field* tertentu dari tabel-tabel bersangkutan. Coba Anda perhatikan tabel PENERBIT dan BUKU yang telah dibuat sebelumnya. Kedua tabel tersebut memiliki relasi melalui *field* KODE_PENERBIT. Dalam kasus semacam ini, tabel PENERBIT disebut sebagai tabel referensi bagi tabel BUKU. Berikut ini contoh sintaks untuk melakukan seleksi dari dua tabel tersebut. Di sini kita akan menggunakan alias untuk menyederhanakan penulisan sintaks.

```
select
a.kode_buku, a.judul, a.pengarang,
b.nama_penerbit, a.jumlah_buku
from
```



```

buku a,
penerbit b
where
a.kode_penerbit = b.kode_penerbit

```

Hasil yang akan diberikan:

kode_buku	judul	pengarang	nama_penerbit	jumlah_buku
K006	36 Mahir Komputer	Santoso	Elex	3
K002	Pemrograman Delphi	Antoni Pranata	Andi	5
K001	Menguasai ORACLE	Budi Raharjo	Informatika	10
K003	Algoritma & Pemrograman	Rinaldi	Informatika	15
K004	Pemrograman PHP	Betha S	Informatika	8
K005	Map Server	Ruslan N	Informatika	2
K007	Mudah Belajar Java	Budi Raharjo	Informatika	15
K008	MS Access	Imam Heryanto	Informatika	20

Aggregate Function

Aggregate function adalah fungsi *built-in* yang sering digunakan dalam proses seleksi data. Bagian ini akan menjelaskan kegunaan fungsi-fungsi tersebut.

MAX

Fungsi MAX digunakan untuk menentukan nilai maksimum dari suatu data yang terdapat pada *field* tertentu.

Bentuk umum:

```
select max(field) from NamaTabel
```

Contoh:

```
select max(harga) from produk;
```

max(harga)
1500000,0

Fungsi MAX juga berlaku untuk menentukan nilai maksimal dari data yang bertipe karakter atau *string*.

MIN

Fungsi MIN digunakan untuk menentukan nilai minimum dari suatu data yang terdapat pada *field* tertentu.

Bentuk umum:

```
select min(field) from NamaTabel
```

Contoh:

```
select min(harga) from produk;
```

min(harga)
250000,0

Sama halnya seperti fungsi MAX, fungsi MIN juga dapat digunakan untuk data dengan tipe karakter atau *string*.

AVG

Fungsi AVG digunakan untuk menentukan nilai rata-rata dari semua data yang terdapat pada suatu *field* tertentu.

Bentuk umum:

```
select avg(field) from NamaTabel
```

Contoh:

```
select avg(harga) from produk;
```

avg(harga)
916666,666667

Fungsi ini hanya berlaku untuk data dengan tipe numerik. Jika diaplikasikan ke tipe karakter, fungsi tersebut akan mengembalikan nilai 0.

COUNT

Fungsi COUNT digunakan untuk menghitung cacah *record* dari suatu *field*.

Bentuk umum:

```
select count(field) from NamaTabel
```

Contoh:

```
select count(*) from produk;
```

count(*)
6

Nilai NULL yang terdapat pada suatu sel di dalam *field* tertentu tidak akan dihitung jika Anda menggunakan fungsi COUNT. Tanda * digunakan untuk mewakili seluruh *field* yang ada pada tabel bersangkutan.

Sum

Fungsi SUM digunakan untuk menjumlahkan data yang terdapat pada suatu *field* tertentu.

Bentuk umum:

```
select sum(field) from NamaTabel
```

Contoh:

```
select sum(harga) from produk;
```

sum(harga)
5500000,0

Fungsi ini hanya dapat digunakan untuk data dengan tipe numerik.

Grouping

Pada saat bekerja dengan *aggregate function*, kita akan (hampir) selalu menggunakan *grouping* (pengelompokkan data). Proses *grouping* dilakukan menggunakan klausa GROUP BY. Untuk melakukan proses ini, silakan Anda ubah terlebih dahulu struktur dan data pada tabel produk menggunakan sintaks berikut:

```
create table vendor (  
    id_vendor char(2) not null,  
    nama_vendor varchar(50),  
    constraint pk_vendor primary key(id_vendor)  
);  
  
alter table produk  
    add column id_vendor char(2);  
  
alter table produk  
    add constraint fk_produk_vendor foreign key(id_vendor)  
        references vendor(id_vendor);  
  
insert into vendor values('01', 'CV. Multi Jaya');  
insert into vendor values('02', 'Toko Muda Elektronika');  
insert into vendor values('03', 'PT. ABC');  
  
update produk  
    set id_vendor = '03'  
    where id = 'EL-001';  
update produk  
    set id_vendor = '01'  
    where id = 'EL-002';  
update produk  
    set id_vendor = '02'
```

```
where id = 'EL-003';
update produk
set id_vendor = '02'
where id = 'EL-004';
update produk
set id_vendor = '03'
where id = 'EL-005';
update produk
set id_vendor = '03'
where id = 'EL-006';
```

Setelah sintaks di atas dieksekusi, kita sudah bisa melakukan *grouping* terhadap tabel `PRODUK` berdasarkan `VENDOR` tertentu. Sebagai contoh, kita ingin menghitung jumlah total berapa kali `vendor` tersebut memasok barang. Sintaks yang diperlukan adalah sebagai berikut:

```
select
  vendor.nama_vendor, count(*) as 'Pemasokan'
from
  produk, vendor
where
  produk.id_vendor = vendor.id_vendor
group by
  vendor.nama_vendor;
```

Hasil yang akan diberikan:

Hasil dari suatu *group* dapat difilter menggunakan klausa `HAVING`. Berikut ini contoh sintaks yang akan menunjukkan hal tersebut.

```
select
  vendor.nama_vendor, count(*) as 'Pemasokan'
from
  produk, vendor
where
  produk.id_vendor = vendor.id_vendor
group by
  vendor.nama_vendor
```

```
having
  count(*) >= 2;
```

Selain itu, hasil dari *group* juga dapat diurutkan berdasarkan *field* tertentu, misalnya seperti yang tampak pada sintaks di bawah ini.

```
select
  vendor.nama_vendor, count(*) as 'Pemasokan'
from
  produk, vendor
where
  produk.id_vendor = vendor.id_vendor
group by
  vendor.nama_vendor
having
  count(*) >= 2
order by
  pemasokan;
```

Daftar Pustaka

- Cosentino, Cristopher. 2002. *Advanced PHP for Web Professionals*, Prentice Hall PTR.
- Lerdorf, Rasmus & Tatro, Kevin. 2002. *Programming PHP*, O'Reilly.
- Meloni, Julie C. 2002. *PHP Fast & Easy Web Development*, Premier Press.
- Meloni, Julie C. 2002. *Sams Teach Yourself PHP, MySQL™ and Apache in 24 Hours*, Sams Publishing.
- Sklar, David. 2004. *Learning PHP 5*, O'Reilly.
- Trachtenberg, Adam. 2004. *Upgrading to PHP 5*, O'Reilly.
- Valade, Janet. 2004. *PHP 5 for Dummies*, Wiley Publishing, Inc.